JOINT VARIATIONAL CAMERA CALIBRATION REFINEMENT AND 4-D STEREO RECONSTRUCTION APPLIED TO OCEANIC SEA STATES

A Thesis
Presented to
The Academic Faculty

by

Ping-Chang Shih

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2014

JOINT VARIATIONAL CAMERA CALIBRATION REFINEMENT AND 4-D STEREO RECONSTRUCTION APPLIED TO OCEANIC SEA STATES

Approved by:

Professor Patricio Vela,
Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Francesco Fedele
School of Electrical and Computer Engineering
School of Civil and Environmental Engineering
Georgia Institute of Technology

Professor Anthony Yezzi, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Justin Romberg
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Allen Tannenbaum
Computer Science Department
Stony Brook University

Professor Frank Dellaert
School of Interactive Computing
Georgia Institute of Technology

Date Approved: June 20, 2014

獻給父親施教修和母親賴鳳英:
為其無止盡且無條件的愛和支持,
無論有無這個學位。

To my parents,

Chiao-Hsiu Shih and Feng-Ying Lai,

for their everlasting and unconditional love and support—with or without this degree.

ACKNOWLEDGEMENTS

In Mandarin, a person who brings tremendously beneficial and perpetual impacts upon our lives is called a "**貴人**." I have met four **貴人** at Georgia Tech during the past eight years of pursuing my Ph.D. degree. They have guided me to cross this treacherous mist, accompanied me through this agonizing journey, and helped and encouraged me when I made my way through this labyrinth. I sincerely acknowledge Dr. Anthony Yezzi, Dr. Francesco Fedele, Dr. Guillermo Gallego, and Mr. Jonathan Ong. I thank them for being my "giants"; standing on their "shoulders," I learn to see farther.

People always say that a Ph.D. student will eventually grow to professionally and characteristically resemble his or her advisor. If this is the case, being the student of Dr. Yezzi is undoubtedly the most promising commencement of my technical career. I admire his knowledge, generosity, and personality. Working with Dr. Yezzi is the greatest happiness I have ever enjoyed as a student.

Full of enthusiasm and optimism for research, Dr. Fedele, my co-advisor, has been my research "propeller" during the years I work with him. He has been encouraging me to overcome the difficulties I encountered in the ocean wave reconstruction project—the research project I delight in and suffer from. His trust and patience are the main factor that I can unleash my potential to develop myself.

Although officially my labmate, Dr. Gallego is effectively my third advisor. He taught me the first numerical solutions of PDEs; he demonstrated to me the first C++ inheritance program. More importantly, he has been illustrating to me the attitude toward research: learn broadly, think thoroughly, digest systematically, and ruminate frequently.

Mr. Ong served as my personal English teacher since 2008. He spent numerous hours helping me correct and practice my English—from pronunciation to writing—with great patience. If it weren't for his assistance, I could not have grown confident in communicating with others in daily or technical conversations, and my life would have been entirely different from what it is now.

Finally, I also thank Dr. Chi-Ti Hsieh for taking care of me when I first arrived at US, Dr. Yong-Dian Jian for sharing with me the concepts of programming, and Dr. Zhenwu Shi for sharing with me precious experience of job-hunting. I feel sincere gratitude to them for anything they have ever done for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In this thesis, an innovative algorithm for improving the accuracy of variational space-time stereoscopic reconstruction of ocean surfaces is presented. The space-time reconstruction method, developed based on stereo computer vision principles and variational optimization theory, takes videos captured by synchronized cameras as inputs and produces the shape and superficial pattern of an overlapped region of interest as outputs. These outputs are designed to be the minimizers of the variational optimization framework and are dependent on the estimation of the camera parameters. Therefore, from the perspective of computer vision, the proposed algorithm adjusts the estimation of camera parameters to lower the disagreement between the reconstruction and 2-D camera recordings. From a mathematical perspective, since the minimizers of the variational framework are determined by a set of partial differential equations (PDEs), the algorithm modifies the coefficients of the PDEs based on the current numerical solutions to reduce the minimum of the optimization framework. Our algorithm increases the tolerance to the errors of camera parameters, so the joint operations of our algorithm and the variational reconstruction method can generate accurate space-time models even using videos captured by perturbed cameras as input. This breakthrough prompts the realization of ocean surface reconstruction using videos filmed by remotely-controlled helicopters in the future. A number of techniques, technical or theoretical, are explored to fulfill the development and implementation of the algorithm and relative computation issues. The effectiveness of the proposed algorithm is validated through the statistics applied to real ocean surface reconstructions of data collected from an offshore platform at the Crimean Peninsula, the Black Sea. Moreover, synthetic data generated using

computer graphics are customized to simulate various situations that are not recorded in the Crimea dataset for the demonstration of the algorithm.

CHAPTER I

INTRODUCTION

Studies of wave climate, extreme ocean events, turbulence, and the energy dissipation of breaking and non-breaking waves are closely related to the measurements of the ocean surface. To gauge and analyze ocean waves on a computer, we reconstruct their 3-D or 4-D model by utilizing the concepts of stereoscopic reconstruction and variational optimization. This technique requires at least two calibrated cameras—cameras whose parameters are estimated for the mathematical projection from space to an image plane—to take videos of the ocean surface as input. However, the accuracy of camera parameters, including the orientations and the positions of cameras as well as the internal specifications of optics elements, are subject to environmental factors and manual calibration errors. These errors of the camera parameters magnify the errors of the 3-D or 4-D variational reconstruction after projection, so to improve the accuracy of the reconstruction, we should remove the errors introduced to the camera parameters. In this dissertation, we conceive algorithms to refine the camera parameters for the 3-D and 4-D (space-time) reconstructions of the Variational Wave Acquisition Stereo System (VWASS) project, in which the variational approach is used for the reconstruction the ocean surface. Our algorithms are designed based on the mathematical architecture of the VWASS [19, 21]; therefore, our algorithms can be seamlessly integrated to the VWASS to jointly refine the camera parameters and generate improved reconstructions.

1.1 Origin and history of the problem

Artificially reproducing ocean surfaces and related phenomena has been an extremely challenging task among different disciplines. The movements of ocean surfaces are

fast and their changes are affected by natural and local factors, such as gravity, winds, currents, and even the passes of ships, so the replicas of ocean surfaces in controlled environments are difficult to be obtained. As a result, to validate theory proposed to describe or estimate ocean surfaces, scientists have been dedicating in efforts to record the states of ocean surfaces in the field using various sensors.

Different types of sensors produce different types of data: Floating sensors with one end attached to the seabed (the most tradiational means) can produce the temporal height changes of some spots of interest. However, this type of data are usually 1-D and are consequently not informative for wave analyses. Satellites and radars can be used to measure ocean waves (2-D data), such as the case in [31]. However, the costs of using satellites or radars are too high for small research labs to afford, and the usage of the apparatus is usually limited by weather conditions. In addition, the resolution of the data outputs are usually in the unit of kilometers, which may be inapplicable to gauging an ocean surface with the range of meters.

With the decreasing prices and advancing functions of cameras, photography has emerged to substitute for geodesic observations of ocean surfaces. Back to 1975, Sugimori [45] utilized vertical aerial photographs to study the directional distortion of wave spectra. In the 1980's, Holthuijsen [25] adopted stereo photography and synchronous images captured by cameras installed on helicopters to explore the directional energy distribution of waves in fetch-limited conditions. However, the accuracy of the outputs in the era were not high enough given that relative photography techniques were not mature from the standpoint of nowadays.

Within the past twenty years, computer vision principles have been gradually adopted to create three-dimensional (3-D) models of the ocean surface for measurement and analysis purposes. These computer models are generated through a process known as 3-D reconstruction in the field of stereo computer vision, where the 3-D shape of an object is recovered based on its 2-D projections (e.g. observed images)

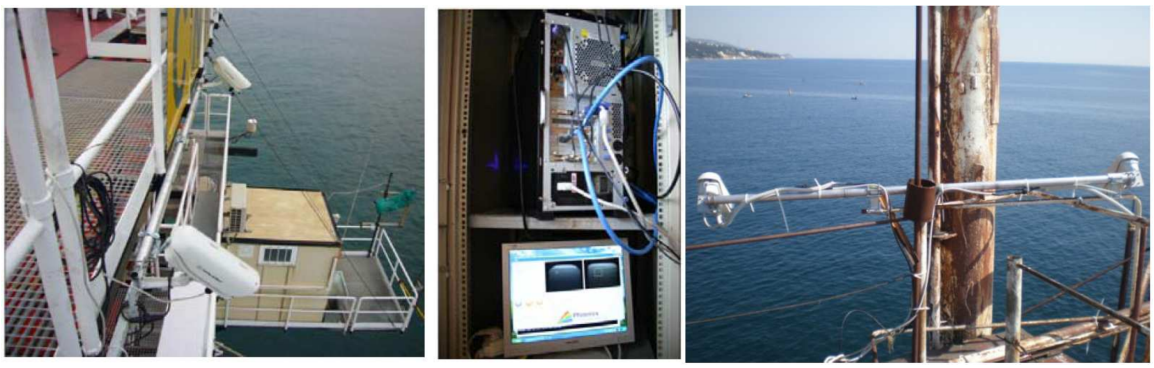


Figure 1: Acqua Alta, the oceanographic tower located in the Northern Adriatic Sea. On the platform, cameras are synchronized and oriented to film an overlapped region of ocean surface for the 3-D reconstruction of waves.

obtained from various viewpoints and the parameters of the viewpoints. Figure 1 exhibits commom 3-D reconstruction equipment of ocean waves: In the equipment, the cameras are installed on an offshore platform, so the synchronization between cameras is no longer a technical issue since cameras can be controlled by off-the-shelf programs on computers, which benefits the acquirement of the data and the accuracy of the measurement. Moreover, because the distance between cameras is a crucial parameter for the reconstruction and is fixed on the platform, compared to [25], the outputs of the equipment in Figure 1 usually have higher precision. Note that because the distance between the cameras and the ocean surface is usually less than twenty meters, the scale of the outputs of this type method is therefore of the range of several meters in dimension.

In detail, 3-D reconstruction methods can be categorized into two types: epipolar and variational methods. Due to its simplicity, the former type was preferred by many researchers to reconstruct ocean surfaces [2, 3, 12, 33, 41, 42, 49]. Such methods achieve reconstruction in two steps: by first matching corresponding features (e.g. points) across images and then by back-projecting them to produce a set of 3-D points (e.g. a point cloud) that represents the spatial positions of the observed 2-D features. However, this type of methods is not particularly effective when being

applied to a target object that lacks distinctive (e.g. textured) image features, such as the ocean surface. Consequently, the yielded point cloud representation of the object surface is sparse and therefore inadequate for applications requiring a dense model.

Variational 3-D reconstruction methods have been founded upon the advantages of the calculus of variations to overcome the aforementioned disadvantages of the feature-based methods. By converting a computer vision problem into a variational optimization one, these 3-D reconstruction methods approximate the surface of the target object by piece-wise smooth functions [13, 51] instead of a collection of 3-D points. Therefore, users can arbitrarily sample the piece-wise smooth functions to visualize the reconstruction at any resolution or to analyze the reconstruction at any location. Utilizing this concept, Gallego *et al.* [19, 21, 22] proposed a variational framework to reconstruct the 3-D model of a patch of ocean surface and extended the concept to the reconstruction the space-time model (4-D). In this framework, the reconstructed surface of the object is obtained as the minimizer of a functional that takes into account both the spatial (or spatial and temporal, in the 4-D case) smoothness of the surface and its photometric error. The latter quantifies the discrepancies between the measurements (snapshots of videos) and the "reprojections" of the reconstruction onto measurements via mathematical coordinate transformations and pin-hole projection formulas. The resulting reconstruction consists of two parts: the height or elevation map and the superficial texture pattern (called radiance map) of the ocean surface. When the radiance map is rendered on top of the corresponding elevation map, a computer model of the ocean surface is obtained.

However, the accuracy of all aforementioned computer-vision-generated models, through epipolar or variational methods, strongly depends on the accuracy in the determination of the cameras' parameters, a technique called camera calibration. These parameters specify the perspective projection operation carried out by the cameras

(mapping points in the 3-D world to 2-D domains on sensors) and its inverse operation (back-projection), hence they have a direct effect in the evaluation of photometric errors (or called reprojection errors). The reconstruction process is very sensitive to the changes of camera parameters, e.g., small deviations of the camera parameters can cause a magnified incorrect reprojection error and, consequently, change the minimizers of the error functional, yielding incorrect reconstruction. Since cameras are installed outdoors (often on an offshore platform) in our main application, extrinsic camera parameters—parameters that accounts for the relative orientation and location of the cameras in the scene—are prone to be corrupted by natural factors such as breeze or vibrations. By contrast, even though intrinsic parameters—parameters specifying dimensions of optics elements—may remain constant, they are likely to carry errors introduced during a camera calibration procedure. Therefore, the reconstruction of a patch of the ocean surface over a time interval should not be performed alone but be incorporated with a camera calibration refinement technique.

1.2 Research objectives

We address the problem of reconstructing a patch of the ocean surface over a given period of time during which extrinsic camera parameters might be perturbed by environmental factors and intrinsic camera parameters might not be accurately estimated. To this end, we incoporate a camera calibration refinement method into the variational optimization framework developed in [19, 21, 22] to jointly perform the 4-D reconstruction of ocean surfaces and the refinement of the camera parameters. They are formulated as the minimizers of a specially designed error functional that consists of 1) the discrepancies between acquired videos and the reprojection of the 4-D reconstruction onto the videos, 2) the spatial and temporal characteristics of ocean surfaces, and 3) the temporal variance of the camera parameters. As a result of iteratively obtaining the minimizers of the error functional, our algorithm can reconstruct

the space-time model for the target region and decrease the influence of the errors caused by deviations of the camera parameters on the reconstruction.

The presented algorithm and the demonstrated experiements in this thesis would be promising evidence of the feasibility of determining the 4-D ocean surfaces from video recordings using variational stereoscopic methods. After all, reducing the influence arising from the camera parameter deviations is a crucial factor that prompts offshore or oil industries to adopt similar technology as a means of studying oceanography in the future.

1.3 Organizational structure of this thesis

This thesis is composed of seven chapters, which are arranged according to the chronological development of each technical part. The first chapter covers the introduction to this project. Chapter 7 gives an overall conclusion of the project. Chapter 2 covers the related theoretical background of understanding this thesis. After that, we step into technical details, including theory and implementation, of how the calibration refinement of the variational stereo computer vision problem is formulated into an optimization one and how it is solved. This step involves in several long series of derivations, and most of them are listed in appendix sections. The ultimate goal of this project—joint camera calibration refinement and 4-D reconstruction of ocean surfaces—is demonstrated in chapter 5 and chapter 6. Before them, we inserted chapter 4, in which a novel method integrating computer graphics and stereo computer vision is developed to generate simulation data for validating our algorithm.

CHAPTER II

VARIATIONAL 3-D RECONSTRUCTION OF SEA STATES

2.1 Background knowledge

In the modern stereo computer vision and 3-D reconstruction fields, the procedure of acquiring extrinsic and intrinsic parameters of the camera projection model is referred to as camera calibration (or camera resectioning). Extrinsic parameters construct linear coordinate transformations between imaginary coordinate systems in which the spatial position of every point can be located. By contrast, intrinsic parameters are internal specifications of cameras that consist of the geometric properties of the optics hardware elements in cameras.

In the camera projection model, 3-D points are generally defined in a universal coordinate system, the world coordinate frame, and then linearly transformed into the coordinate system of each camera. The transformed values then undergo the pin-hole projection, which maps the values onto image planes. Since images carry only 2-D information, some portion of the 3-D information is lost, suggesting that recovering 3-D information based on only one image is impossible. Hence, most 3-D reconstruction algorithms work as the reverse process of the camera projection model: taking multiple images as inputs to recover 3-D structures.

The accuracy of a 3-D reconstruction algorithm depends on the estimations of the extrinsic and intrinsic parameters, which can be obtained prior to the 3-D reconstruction process. In the computer vision community, the type of 3-D reconstruction in which the camera parameters are obtained prior to the 3-D reconstruction process is referred to as "calibrated 3-D reconstruction." Calibrated 3-D reconstruction

methods are more accurate than uncalibrated ones, in which the parameters and 3-D reconstruction are processed simultaneously, so we will focus on only the topics relevant to calibrated reconstruction.

The most popular open-source camera calibration tools are Camera Calibration Toolbox for MATLAB [6] and OpenCV [7]. To determine the camera parameters, one needs to take pictures of a planar pattern with sets of parallel lines, such as a chessboard pattern. Parallel lines in the 3-D world intersect at points in infinity (called "vanishing points"), but after experiencing the projection effect onto an image, vanishing points move from infinity to finite ranges on the image plane. During the process of locating the vanishing points in the finite range and restoring them back to infinity, the tools obtain necessary information to calculate the extrinsic and intrinsic parameters. However, these parameters are never exactly accurate because of the existence of: 1) noise on the sensors, 2) manual operation errors when users select parallel lines, 3) numerical errors during the calculation of camera parameters, and 4) natural factors such as breeze or vibrations. To determine the fully accurate camera parameters and to obtain an accurate 3-D reconstruction, we need camera calibration refinement, the topic of this dissertation.

2.2 Notation of coordinate transformations in variational 3-D reconstruction

In a standard space-time (3-D) reconstruction setup (see Figure 2), a smooth surface ($S(\boldsymbol{u})$) and each 3-D point on it ($\mathbf{X}(\boldsymbol{u}) = \big(X(\boldsymbol{u}), Y(\boldsymbol{u}), Z(\boldsymbol{u})\big)^{\mathsf{T}}$) are represented as the function values of a domain[1] in the world coordinate system. Alternatively, these 3-D points can be depicted in the i^{th} camera coordinate system by the following notation: $\tilde{\mathbf{X}}_i = (\tilde{\mathsf{X}}_i, \tilde{\mathsf{Y}}_i, \tilde{\mathsf{Z}}_i)^{\mathsf{T}}$. $\mathbf{X}$ and $\tilde{\mathbf{X}}_i$ are related in the following linear transformation:

$$\tilde{\mathbf{X}}_i = \mathsf{R}_i\mathbf{X} + \mathbf{t}_i. \tag{1}$$

[1]This domain is a 2-D parameter space, U, in which each point is denoted by $\boldsymbol{u}$ ($\boldsymbol{u} = (u, v)^{\mathsf{T}}$).

In equation (1), R_i is a three-by-three rotation matrix accounting for the difference between the orientations of the world coordinate and the i^{th} camera coordinate systems, and the displacement between the origins of the two systems is $\mathbf{t}_i$, a three-by-one vector.

In the pin-hole camera model, each 3-D point $\tilde{\mathbf{X}}_i$ is projected onto the image plane, forming a 2-D point, $\mathbf{x}$. This projection process is mathematically represented as $\mathbf{x} = \left(\bar{x}, \bar{y}\right)^{\mathsf{T}} = \left(\frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i}, \frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i}\right)^{\mathsf{T}}$. However, representation $\mathbf{x}$ is a reading in the i^{th} camera coordinate systems, so one more coordinate transformation is required to convert it to pixel indices $\hat{\mathbf{x}} = \left(\hat{x}, \hat{y}\right)^{\mathsf{T}}$, which is how people interpret the position of $\mathbf{x}$ on an image plane of a charge-coupled device (CCD). This transformation is

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = \left(\underbrace{\begin{bmatrix} L_x & a & \zeta_0 \\ 0 & L_y & \eta_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{K}_i} \begin{bmatrix} \tilde{\mathsf{X}}_i \\ \tilde{\mathsf{Y}}_i \\ \tilde{\mathsf{Z}}_i \end{bmatrix} \right)_n = \begin{bmatrix} L_x \frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i} + a \frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \zeta_0 \\ L_y \frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \eta_0 \end{bmatrix}. \tag{2}$$

Subscript n in this equation indicates a non-linear transformation: converting the homogeneous coordinates in the parentheses to Cartesian coordinates (see [23] for the definition of homogeneous coordinates). All parameters in K_i are internal specifications of a CCD camera: $\left(L_x, L_y\right)^{\mathsf{T}}$ is the focal length in pixels, a, usually 0 in most cameras, is the the skew coefficient between $\hat{x}$ and $\hat{y}$, and $\left(\zeta_0, \eta_0\right)^{\mathsf{T}}$ is the location of the principal point, at which axis $\tilde{\mathsf{Z}}_i$ intersects the image plane.

Whereas R_i and $\mathbf{t}_i$ are generally categorized as extrinsic camera parameters, parameters in K_i are categorized as intrinsic camera parameters. All symbols and their relationships are shown in Figures 2 and 3: Figure 2 depicts the geometrical arrangement of all coordinate systems, and Figure 3 presents a flowchart of the algebraic relationships among all the transformations.

Note that when we start to generalize the concept of 3-D reconstruction to 4-D reconstruction in chapter 5, we add superscript τ to the extrinsic parameters of the

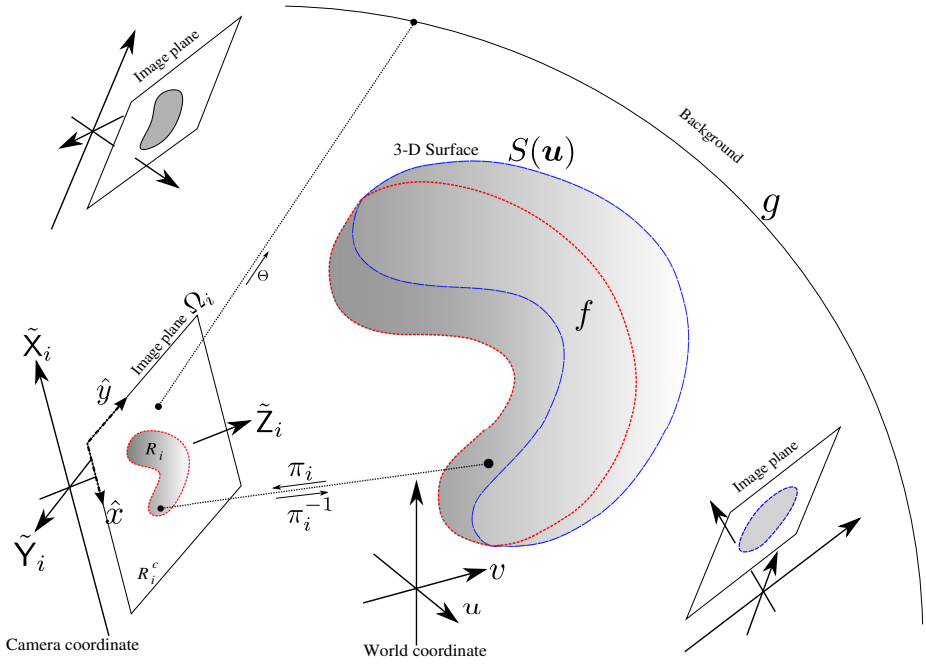


Figure 2: The geometric relationships between coordinate systems. The target object, $S(\boldsymbol{u})$, and the coordinates of any point on $S(\boldsymbol{u})$, $\mathbf{X}(\boldsymbol{u})$, are represented as function values in the world coordinate system. The argument, $\boldsymbol{u}$ ($\boldsymbol{u} = (u, v)^\mathsf{T}$), represents any point in a 2-D parameter space, U. A 3-D point, $\mathbf{X}$, is imaged by pin-hole cameras and then converted to pixel indices, $\hat{\mathbf{x}}$. (Figure courtesy of Dr. Anthony Yezzi in [40])

cameras—R_i^τ and $\mathbf{t}_i^\tau$—to emphasize the temporal moment, τ, at which they are estimated. Intrinsic camera parameters—parameters in K_i—are not particularly marked since we assume that they remain constant at each temporal moment even under the influence of environmental factors or that the effect on the system of their small variations is negligible compared to that of the extrinsic parameters. In addition, for the succinctness issue, τ is not particularly added to some symbols such as $\hat{\mathbf{x}}$, $\mathbf{x}$, and $\tilde{\mathbf{X}}_i$, even though they are dependent on variable τ.

2.3 Stereoscopic segmentation

A generic variational 3-D reconstruction problem can be depicted as the configuration in Figure 2 and interpreted as a stereoscopic segmentation problem [51]. In a 2-D segmentation case, the segmentation can be performed with the concept of active

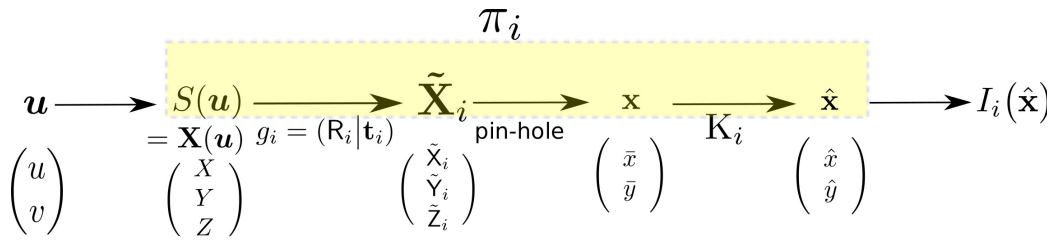


Figure 3: The algebraic relationships between coordinate systems. The forward projection (the highlighted part) is denoted by π_i. π_i maps $\mathbf{X}(\boldsymbol{u})$, the reading in the world coordinate system, to $\hat{\mathbf{x}}$, the reading in pixel indices of the i^{th} camera. Moreover, $I_i(\hat{\mathbf{x}})$ is the pixel intensity of $\hat{\mathbf{x}}$ on the image captured by the i^{th} camera.

contours [10, 11], which evolve based on the 2-D features to separate regions. This concept can be used to perform 3-D reconstruction if 2-D active contours are generalized to 3-D active surfaces. In the concept of stereoscopic segmentation, the object to be reconstructed is an active surface that evolves based on its imaged features on the images. The evolution is designed to stop when the discrepancies between the measurements on the images and reprojections of the scene (including the active surface and the background) is minimized. Because the minimizer is usually obtained when the reprojected occluding boundaries of the active surface fit the visible boundaries of the measurements on the images, this method is treated as a 3-D segmentation and therefore called "stereoscopic segmentation."

The stereoscopic segmentation configuration [51] applies to a scene with a foreground and a background and reconstructs the shape of the foreground, the radiance of the foreground, and the radiance of the background, denoted by S, f, and g, respectively. S, f, and g are piece-wise smooth functions; initially unknown, they eventually evolve to optimal values after error functional (3)[2] is minimized.

$$E(S, f, g) = E_{data}(S, f, g) + E_{smooth}(f, g) + E_{geom}(S), \tag{3}$$

[2]This is also called the Yezzi-Soatto error functional, which is a variation of [37].

where

$$\begin{cases} E_{data} = \sum_{i=1}^{N_c} \left\{ \int_{R_i} \left[I_i(\hat{\mathbf{x}}) - f\left(\pi_i^{-1}(\hat{\mathbf{x}})\right) \right]^2 d\hat{\mathbf{x}} + \int_{R_i^c} \left[I_i(\hat{\mathbf{x}}) - g\left(\Theta(\hat{\mathbf{x}})\right) \right]^2 d\hat{\mathbf{x}} \right\} \\ E_{geom} = \int_S dA \\ E_{smooth} = \int_S \frac{1}{2} \|\nabla f\|^2 dA + \int_B \frac{1}{2} \|\nabla g\|^2 dA \end{cases} . \quad (4)$$

In the equations above, N_c is the number of images and B is the surface of the background. E_{geom} is the surface area of target object S, so the role of E_{geom} in E is to control the smoothness of S. By contrast, the smoothness of f and g is represented as the integrals of the l^2-norms of the corresponding gradients, indicating that E_{smooth} controls the smoothness of f and g. E_{data}, referred to as the data fidelity term, quantifies the discrepancies between the reprojections of the scene and the measurements on all images: the term with integral limit R_i in E_{data} represents the errors between the reprojected foreground (the active surface) and the intensities of image pixels within the reprojection region. Similarly, the other term in E_{data} represents the errors measured outside R_i.

Since the error functional is designed with the calculus of variations and because the minimizers—S, f, and g—are obtained through the solutions of the first variation of the error functional, this 3-D reconstruction, which converts a computer vision problem into an optimization problem, is called "variational 3-D reconstruction." By arithmetically attaining the minimum of equation (3),[3] we obtain a set of S, f, and g that approximate the visual measurements such as shapes and texture on all images.

2.3.1 Variational 3-D reconstruction of ocean waves: constrained graph evolution of stereoscopic segmentation

With the maturation of stereo computer vision principles and the appearances of open source computer vision libraries, using 3-D reconstruction techniques to reconstruct

[3]This procedure is usually performed via iterative solvers of PDEs, which will be discussed in the following section.

ocean waves is becoming popular in civil engineering research fields. Most of these models are produced through epipolar geometry [3, 32, 34, 49], which determines the spatial positions of matched feature points on input images. These spatial positions form a sparse point cloud with insufficient information for applications requiring dense ocean wave models. In addition, each point cloud—built from measurements at each moment—contains sporadic "holes" resulting from unpaired feature points. The holes randomly appear at various positions in a series of point clouds reconstructed from synchronous video sequences, thereby undermining the temporal analyses of wave models in the form of point clouds.

To overcome the aforementioned drawbacks, Gallego [19, 21] modified the stereoscopic segmentation algorithm to apply to the 3-D reconstruction of ocean waves. The modified error functional is

$$E(f, Z) = E_{data}(f, Z) + \alpha E_{geom}(Z) + \beta E_{rad}(f), \tag{5}$$

where $\alpha, \beta > 0$ and

$$\begin{cases} E_{data} = \sum_{i=1}^{N_c} \left\{ \int_{R_i} \frac{1}{2} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}} \right\} \\ E_{geom} = \int_U \frac{1}{2} \|\nabla Z(\boldsymbol{u})\|^2 d\boldsymbol{u} \\ E_{rad} = \int_U \frac{1}{2} \|\nabla f(\boldsymbol{u})\|^2 d\boldsymbol{u} \end{cases} . \tag{6}$$

In the equations above, Z and f are functions of variables (u, v) in domain $\boldsymbol{u}$. Z and f are called the elevation map and the radiance map, respectively. Recall that in the scenario of stereoscopic segmentation, both the foreground and the background are assumed to exist in the scene and the foreground object is assumed to be a closed surface. Here, the modified error functional defined in equations (5) and (6) applies to an estimated portion of an object surface,[4] generating Z and f to approximate the height and the radiance of a region of an ocean surface, respectively. The roles

[4]This change drops term $\int_{R_i^c} \Big[I_i\big(\hat{\mathbf{x}}\big) - g\big(\Theta(\hat{\mathbf{x}})\big) \Big]^2 d\hat{\mathbf{x}}$ in E_{data} of equation (4).

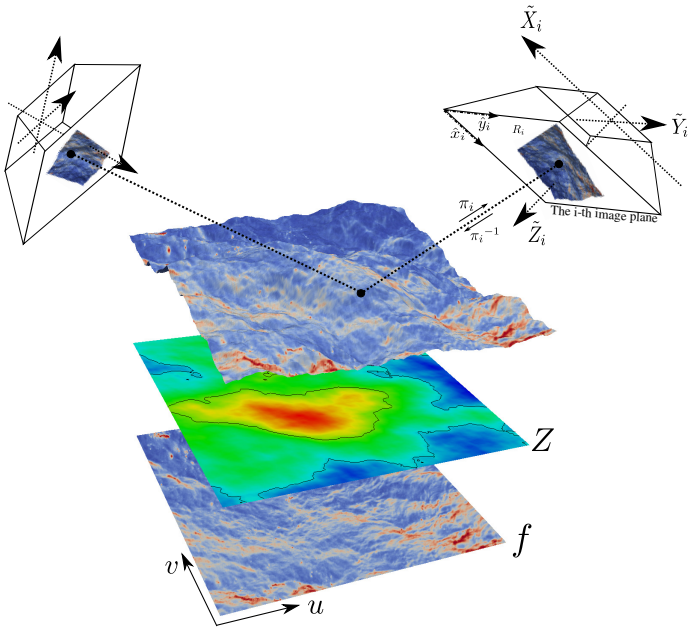


Figure 4: The illustration of the elevation map (Z), the radiance map (f), coordinate system configuration, and projections. Z and f, determined as the minimizers of the variational framework, are rendered together (the graph on the top) to form the reconstruction model of the region of interest.

of Z and f for the reconstruction of ocean surfaces are illustrated in Figure 4: The reconstruction is determined when each coordinate of $\big(u, v, Z(u, v)\big)$ is tinted with color value $\big(u, v, f(u, v)\big)$. In other words, the rendering of f on Z is the reconstruction of the observed region.

From a mathematical perspective, the elevation and radiance maps—the minimizers of an error functional that measures the reprojection error and the smoothness of the model—are determined by the zero first variation of equation (5), forming the following PDEs:

$$g(Z, f) - \alpha \Delta Z = 0 \quad \text{in } U, \tag{7}$$

$$-\sum_{i=1}^{N_c} (I_i - f) J_i(Z) - \beta \Delta f = 0 \quad \text{in } U, \tag{8}$$

$$b(Z, f) + \alpha \frac{\partial Z}{\partial \boldsymbol{\nu}} = 0 \quad \text{on } \partial U, \tag{9}$$

$$\beta \frac{\partial f}{\partial \boldsymbol{\nu}} = 0 \quad \text{on } \partial U, \tag{10}$$

where $g(Z, f)$ and $b(Z, f)$ are

$$g(Z, f) = \nabla f \cdot \sum_{i=1}^{N_c} |\mathrm{K}_i| \tilde{\mathsf{Z}}_i^{-3} (I_i - f)(u - C_i^1, v - C_i^2), \tag{11}$$

$$b(Z, f) = \sum_{i=1}^{N_c} \frac{1}{2} (I_i - f)^2 |\mathrm{K}_i| \tilde{\mathsf{Z}}_i^{-3} \big((u - C_i^1)\nu^u + (v - C_i^2)\nu^v \big). \tag{12}$$

In the equations above, J_i is the Jacobian of the change of variables from the world coordinate system to the i^{th} camera coordinate system. Based on the derivations in [21], J_i is

$$J_i = \frac{|\mathrm{K}_i|}{\tilde{\mathsf{Z}}_i^3} max \left(0, -\langle \mathbf{X} - \boldsymbol{C}_i, \mathbf{X}_u \times \mathbf{X}_v \rangle\right),$$

in which $\boldsymbol{C}_i$ ($\boldsymbol{C}_i = \left(C_i^1, C_i^2, C_i^3\right)^{\mathsf{T}}$) is the coordinates of the camera center of the i^{th} camera located in the world coordinate system.

CHAPTER III

VARIATIONAL CAMERA CALIBRATION REFINEMENT

3.1 Constrained error function

The numerical solutions of equations (7)–(12) are dependent on $\tilde{\mathsf{Z}}_i$ and variables in K_i and $\boldsymbol{C}_i$, which are determined by the camera parameters (intrinsic and extrinsic) introduced in π_i^{-1} in E_{data} of the error functional (5). π_i^{-1}, the notation abuse of π_i, indicates the reverse mapping of π_i, and π_i is an analytical function that maps points from the reconstructed model onto image I_i. With the effect of π_i, any tiny deviation of the camera parameters will introduce a magnified incorrect reprojection error whose minimizers do not approximate the correct shape and radiance and, consequently, produce an incorrect 3-D reconstruction.

Refining the camera parameters for the variational reconstruction of a patch of ocean surface is challenging. The reprojected model within each image domain occupies only a subregion whose size depends upon the camera parameters, so arbitrarily minimizing reprojection error E_{data} in equation (6) may encourage an incorrect refinement of the calibration parameters: The parameters may[1] be tuned to values that shrink R_i (the area of the reprojected model within the image), thereby artificially reducing the reprojection error, E_{data}, to zero. This undesirable process is demonstrated in Figure 3.1 in terms of the reprojections of the reconstruction onto images: In subfigure 5(b), calibration refinement is performed through the minimization of E_{data} with respect to the camera parameters. By contrast, subfigure 5(a) exhibits a

[1]This situation depends upon the feature distribution on the superficial pattern (texture, form the point of view of computer graphics) of the region of interest. In some cases (depending on the texture of the region of interest), arbitrarily minimizing E_{data} from some initial estimates of the camera parameters may NOT lead to zero reprojection area if the relationship between E_{data} and the camera parameter changes is not a monotonically decreasing function.

reconstruction case without any calibration refinement. The areas of the reprojections in subfigure 5(a) are apparently larger than the counterparts in subfigure 5(b) because minimizing E_{data} without any constraints tends to shrink R_i. In addition, subfigure 5(c) indicates that the superficial patterns of the region of interest give rise to a monotonically decreasing relationship between E_{data} and the iterations of tuning the camera parameters.

To overcome this problem, inspired by Unal [48], we propose an algorithm that refines the camera parameters during the reconstruction process of a patch of ocean surface, in which the straightforward reprojection error no longer serves as the basis of optimization.

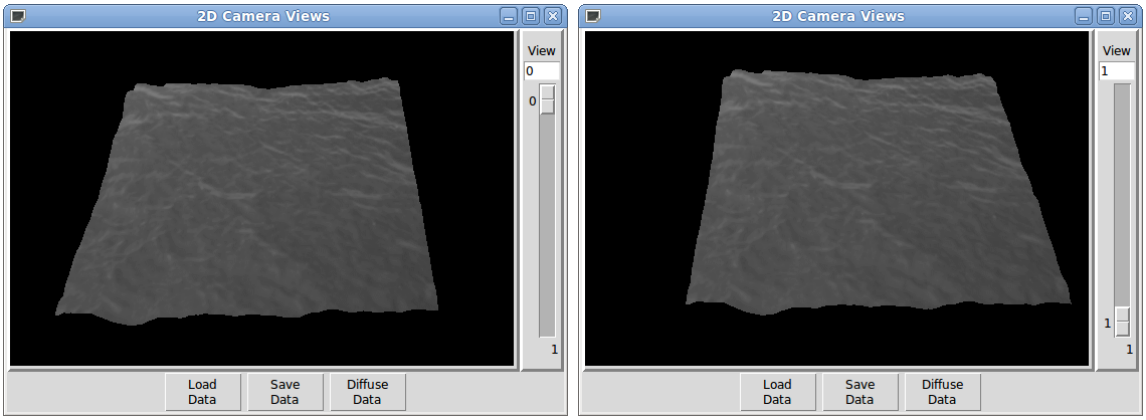
Proposed error function. The designed error function is the product of $\int_{R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big)\Big]^2 d\hat{\mathbf{x}}$ and penalty function $\frac{1}{A_{R_i}}$. Denoted by $E_{data,i}$ to distinguish its connection and difference with E_{data} in equation (5), the error function is defined as

$$E_{data,i} = \frac{\int_{R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big)\Big]^2 d\hat{\mathbf{x}}}{A_{R_i}}, \tag{13}$$

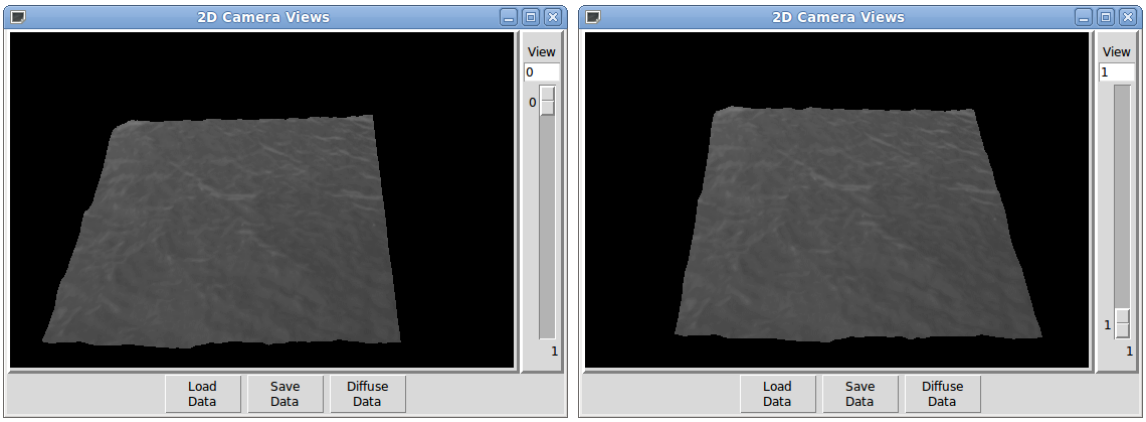
where A_{R_i} is the area of R_i.

The numerator, shown in the braces of ${E_{data}}^2$ of equation (6), has minimum 0 when R_i approaches 0, which could occur when the magnitude of $\mathbf{t}_i$ grows to infinity. That is, if the error function contains only the numerator, arbitrarily minimizing the error function will probably lead to an incorrect estimation of $\mathbf{t}_i$. Hence, to maintain the area of the reprojected model onto the images while the camera parameters are adjusted to diminish E_{data}, we add A_{R_i} to penalize the change of R_i.

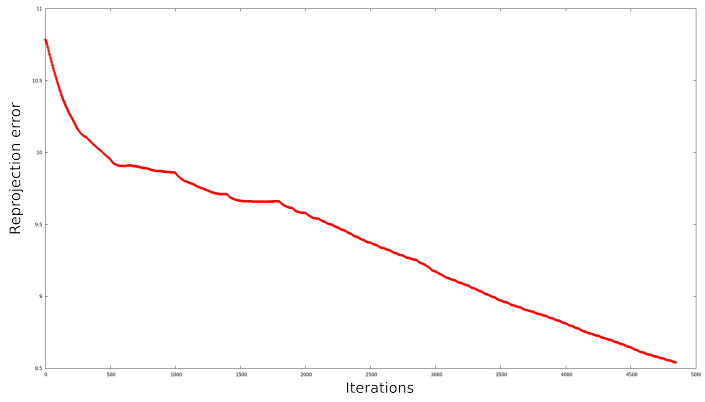
[2]To simplify the derivations, we remove coefficient $\frac{1}{2}$ in the original version, which does not influence the effect of the numerator in equation (13).



(a) The reprojections of the reconstructed model onto image planes (two cameras). In this case, the variational 3-D reconstruction is performed alone without any calibration refinement method.



(b) The reprojections of the reconstructed model onto the image planes. In this case, the reconstruction is accompanied by the calibration refinement method implemented through the minimization of E_{data} with respect to the camera parameters.



(c) The relationship between E_{data} and the iterations of tuning the camera parameters.

Figure 5: Figures 5(a) and 5(b) illustrate the areas of the reprojections onto the image planes. Figure 5(c) demonstrates that E_{data} is inclined to decrease to its global minimum—zero—when E_{data} is arbitrarily minimized with respect to the camera parameters to achieve the calibration refinement.

3.2 Gradient descent flow with respect to the camera parameters

By denoting the parameter(s) under discussion λ[3] and assuming λ is dependent on artificial time variable t, we can obtain a local minimum of $E_{data,i}$ by gradually evolving λ along the gradient descent direction. By the chain rule in calculus,

$$\frac{dE_{data,i}}{dt} = \frac{\partial E_{data,i}}{\partial \lambda}\frac{d\lambda}{dt} = (\nabla_\lambda E_{data,i})^\mathsf{T}\frac{d\lambda}{dt} = \left\langle \nabla_\lambda E_{data,i}, \frac{d\lambda}{dt} \right\rangle_{l^2},$$

where $\langle \cdot, \cdot \rangle_{l^2}$ represents the l^2 inner product operator between two vectors. To find the gradient descent vector, we set the update of λ (denoted by $\Delta\lambda$) to be the negative direction of $\nabla_\lambda E_{data,i}$, yielding

$$\frac{\Delta\lambda}{\Delta t} \equiv -\nabla_\lambda E_{data,i} = -(\frac{\partial E_{data,i}}{\partial \lambda})^\mathsf{T}.$$

Again, after a series of simplifications with the chain rule (see appendix A for details), the derivative of $E_{data,i}$ with respect to λ is

$$\begin{aligned}\frac{\partial E_{data,i}}{\partial \lambda} =& \Bigg\{ - A_{R_i}^{-2}\frac{\partial A_{R_i}}{\partial \lambda}\int_{R_i}\Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big)\Big]^2 d\hat{\mathbf{x}} \\ &+ \frac{2}{A_{R_i}}\int_{R_i} w_i\Big[w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i - I_i\big(\hat{\mathbf{x}}\big)\Big]\left\langle \nabla_{\mathbf{s}} f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big), \frac{\partial \pi_i^{-1}(\hat{\mathbf{x}})}{\partial \lambda}\right\rangle d\hat{\mathbf{x}} \\ &+ \frac{1}{A_{R_i}}\int_{\partial R_i}\Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big)\Big]^2\left\langle \frac{\partial \hat{\mathsf{c}}}{\partial \lambda}, \hat{\mathsf{n}}\right\rangle d\hat{\mathsf{s}}\Bigg\}.\end{aligned}$$

Because π_i^{-1} has no analytical form (recall that π_i^{-1} is just the notation abuse of π_i, an analytical function), we attempt to convert $\frac{\partial \pi_i^{-1}(\hat{\mathbf{x}})}{\partial \lambda}$ to $\frac{\partial \pi_i(\hat{\mathbf{x}})}{\partial \lambda}$ to get a further simplification. By using a change of variables, we lift the region of the integral from

[3] λ can be a scalar or a vector.

Table 1: $\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda}$ and $\frac{\partial(\mathbf{X})}{\partial\lambda}$ with respect to different types of camera parameters

	$\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda}$	$\frac{\partial(\mathbf{X})}{\partial\lambda}$
$\lambda = \left(L_x, L_y\right)^{\mathsf{T}}$	$\frac{1}{\tilde{\mathsf{Z}}_i}\begin{bmatrix}\tilde{\mathsf{X}}_i & 0\\ 0 & \tilde{\mathsf{Y}}_i\end{bmatrix}$	$\frac{1}{\tilde{\mathsf{Z}}_i\lvert\Phi\rvert}\mathsf{R}_i^{-1}\begin{bmatrix}\tilde{\mathsf{X}}_i(\boldsymbol{N}_i\times\vec{\phi_2}) & \tilde{\mathsf{Y}}_i(\vec{\phi_1}\times\boldsymbol{N}_i)\end{bmatrix}$
$\lambda = \left(\zeta_0, \eta_0\right)^{\mathsf{T}}$	$\mathbf{I}$	$\frac{1}{\lvert\Phi\rvert}\mathsf{R}_i^{-1}\begin{bmatrix}(\boldsymbol{N}_i\times\vec{\phi_2}) & (\vec{\phi_1}\times\boldsymbol{N}_i)\end{bmatrix}$
$\lambda = \mathbf{t}_i$	$\boldsymbol{\phi}$	$-\mathsf{R}_i^{-1}\left(\mathbf{I}-\frac{\tilde{\mathbf{X}}_i\otimes\boldsymbol{N}_i}{\langle\tilde{\mathbf{X}}_i,\boldsymbol{N}_i\rangle}\right)$
$\lambda = \vec{\omega}_i$	$-\boldsymbol{\phi}\mathsf{R}_i[\mathbf{X}]_{\mathtt{x}}\Psi$	$\mathsf{R}_i^{-1}\left(\mathbf{I}-\frac{\tilde{\mathbf{X}}_i\otimes\boldsymbol{N}_i}{\langle\tilde{\mathbf{X}}_i,\boldsymbol{N}_i\rangle}\right)\mathsf{R}_i[\mathbf{X}]_{\mathtt{x}}\Psi$

the image domain to world coordinate system U, yielding

$$\frac{\partial E_{data,i}}{\partial\lambda} = \left\{\frac{2}{A_{R_i}}\int_U w_i\Big[w_i f\big(\mathbf{X}\big)+\gamma_i-I_i\big(\pi_i(\mathbf{X})\big)\Big]\left\langle\nabla_{\mathfrak{s}} f\big(\mathbf{X}\big),\frac{\partial(\mathbf{X})}{\partial\lambda}\right\rangle J_i d\boldsymbol{u}\right. \tag{14}$$

$$- A_{R_i}^{-2}\int_U\Big[I_i\big(\pi_i(\mathbf{X})\big)-\big(w_i f\big(\mathbf{X}\big)+\gamma_i\big)\Big]^2 J_i d\boldsymbol{u}\int_{\partial U}\left\langle Q\frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial s},\frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial\lambda}\right\rangle ds \tag{15}$$

$$\left.+\ \frac{1}{A_{R_i}}\int_{\partial U}\Big[I_i\big(\pi_i(\mathbf{X})\big)-\big(w_i f\big(\mathbf{X}\big)+\gamma_i\big)\Big]^2\left\langle Q\frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial s},\frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial\lambda}\right\rangle ds\right\}. \tag{16}$$

In the equations above, ∂U is the boundary of U.

In Terms (14)–(16), only $\frac{\partial(\mathbf{X})}{\partial\lambda}$ and $\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda}$ depend on λ and require further discussion. The details of the derivations of $\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda}$ and $\frac{\partial(\mathbf{X})}{\partial\lambda}$ are listed in appendix A. We summarized the derivation results in Table 1 and listed the symbol explanations in Table 2.

3.3 Baseline constraint

The purpose of penalty function $\frac{1}{A_{R_i}}$ in equation (13) is to maintain the reprojection area of the model while reprojection error $\int_{R_i}\Big[I_i\big(\hat{\mathbf{x}}\big)-\big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big)+\gamma_i\big)\Big]^2 d\hat{\mathbf{x}}$ is being minimized. However, as shown in appendix A, the terms related to A_{R_i} are computationally expensive because they require numerical integrals along ∂R_i, the boundary of R_i.

To replace $\frac{1}{A_{R_i}}$ with other constraints that has a similar effect, we recall that

Table 2: Symbols used in Table 1

$\boldsymbol{N}$:	an unit normal observed in the world coordinate frame
$\boldsymbol{N}_i$:	the observation of unit normal $\boldsymbol{N}$ in the i^{th} camera coordinate frame
$\mathbf{I}$:	an identity matrix. $\mathbf{I}$ can be a three-by-three or two-by-two matrix
$\boldsymbol{\phi}$:	$\begin{pmatrix} L_x & a \\ 0 & L_y \end{pmatrix} \begin{pmatrix} \frac{1}{\tilde{\mathsf{Z}}_i} & 0 & -\frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i^2} \\ 0 & \frac{1}{\tilde{\mathsf{Z}}_i} & -\frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i^2} \end{pmatrix}$
$\vec{\phi_1}$:	the first row of $\boldsymbol{\phi}$
$\vec{\phi_2}$:	the second row of $\boldsymbol{\phi}$
Φ:	$\begin{bmatrix} \boldsymbol{\phi}^{\mathsf{T}} & \mathbf{N}_i \end{bmatrix}^{\mathsf{T}}$
$[\mathsf{a}]_{\mathbf{x}}$:	$\begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$ given $\mathsf{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^{\mathsf{T}}$
$\vec{\omega}_i$:	the rotation axis of R_i
$\bigotimes$:	the outer product operator of vectors
Ψ:	$\frac{\left(\vec{\omega}_i\vec{\omega}_i^{\mathsf{T}} + (\mathsf{R}_i^{\mathsf{T}} - \mathbf{I})[\vec{\omega}_i]_{\mathbf{x}}\right)}{\|\vec{\omega}_i\|^2}$

the purpose of penalty function $\frac{1}{A_{R_i}}$ in $E_{data,i}$ is to prevent A_{R_i} from decreasing and thereby restricting the distances between the ocean surface and the cameras from growing. To this end, we can also enforce a computationally-cheap constraint on the changes of the distances between the ocean surface and the cameras (referred to as depths); inspired by [36], we can even impose a constraint on the distances between cameras (called baselines of cameras) because the length of the baselines is proportional to the depths. Based on this idea, we can replace $E_{data,i}$ with a new cost function E_{calib} designed as follows:

$$E_{calib} = \sum_{i=1}^{N_c} \int_{R_i} \left[I_i(\hat{\mathbf{x}}) - \left(w_i f\left(\pi_i^{-1}(\hat{\mathbf{x}})\right)\right) + \gamma_i \right]^2 d\hat{\mathbf{x}} + \frac{W}{2} \sum_{i=1}^{N_c-1} \sum_{j=i+1}^{N_c} (d_{ij} - d_{ij}^o)^2, \quad (17)$$

where W ($W > 0$) is a weight, d_{ij}^o is the exact distance between the centers of the i^{th} and the j^{th} cameras, and d_{ij} is the estimation of d_{ij}^o during the calibration refinement procedure. Notice that d_{ij}^o is used as a reference to prevent d_{ij} from growing, so d_{ij}^o can be obtained from a calibration refinement tool such as [6, 7] (preferable) or approximated by a measuring tape (in the worst case scenario).

Compared to $E_{data,i}$ defined in equation (13), the adoption of baseline constraints has several advantages, enumerated as follows:

- The calculation of $\sum_{i=1}^{N_c-1}\sum_{j=i+1}^{N_c}(d_{ij}-d_{ij}^o)^2$ requires fewer CPU clock cycles compared to the computation required by A_{R_i} and relative terms.

- The derivative of the second term of equation (17) with respect to any intrinsic camera parameters leads to zero. That is, minimizing E_{calib} to refine the intrinsic parameters requires no computation expense spent on the second term (and the corresponding derivatives) of equation (17). By contrast, computation costs are always necessary for A_{R_i} and its derived term in the case of $E_{data,i}$ when the intrinsic camera parameters are refined.

- The strength of the constraint of E_{calib} can be controlled by weight W. By contrast, no manipulations can be applied to change the effectiveness of A_{R_i} on $\int_{R_i}\Big[I_i\big(\hat{\mathbf{x}}\big)-\big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big)+\gamma_i\big)\Big]^2 d\hat{\mathbf{x}}$.

A big portion of the derivation in appendix A can be reused to obtain the gradient descent flow for minimizing E_{calib}: We can represent d_{ij} in terms of the extrinsic camera parameters, yielding $d_{ij}=\|\mathsf{R}_i^\mathsf{T}\mathbf{t}_i-\mathsf{R}_j^\mathsf{T}\mathbf{t}_j\|$, and equation (17) can be written as

$$E_{calib}=\sum_{i=1}^{N_c}\int_{R_i}\big[I_i(\hat{\mathbf{x}})-\big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big)\big)+\gamma_i\big]^2 d\hat{\mathbf{x}}+\frac{W}{2}\sum_{i=1}^{N_c-1}\sum_{j=i+1}^{N_c}(\|\mathsf{R}_i^\mathsf{T}\mathbf{t}_i-\mathsf{R}_j^\mathsf{T}\mathbf{t}_j\|-d_{ij}^o)^2. \tag{18}$$

By differentiating E_{calib} with respect to the camera parameters of the i^{th} camera, symbol Σ in the first term drops, so $\frac{\partial E_{calib}}{\partial\lambda}$ becomes

$$\frac{\partial}{\partial\lambda}\int_{R_i}\big[I_i(\hat{\mathbf{x}})-\big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big)\big)+\gamma_i\big]^2 d\hat{\mathbf{x}} \tag{19}$$

$$+\quad W\sum_{j=1(j\neq i)}^{N_c}(\|\mathbf{t}_i-\mathsf{R}_i\mathsf{R}_j^\mathsf{T}\mathbf{t}_j\|-d_{ij}^o)\frac{\partial\|\mathbf{t}_i-\mathsf{R}_i\mathsf{R}_j^\mathsf{T}\mathbf{t}_j\|}{\partial\lambda}. \tag{20}$$

Term (19) in different cases can be found in appendix A. Whereas term (20) becomes zero if λ represents the intrinsic parameters, when $\lambda = \mathbf{t}_i$,

$$\frac{\partial \|\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j\|}{\partial \lambda} = \frac{\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j}{\|\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j\|};$$

when $\lambda = \vec{\omega}_i$,

$$\frac{\partial \|\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j\|}{\partial \lambda} = \frac{\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j}{\|\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j\|} \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} [\mathbf{t}_j]_{\mathbf{x}} \mathsf{R}_j \Psi.$$

By using Ω to denote $(\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j)$ for succinctness, we conclude that we can iteratively refine camera parameter λ by adding the current estimation of λ with the following vector:

$$\begin{aligned}
\Delta\lambda &= -\Delta t \Bigg\{ 2 \int_U w_i \Big[w_i f\big(\mathbf{X}\big) + \gamma_i - I_i\big(\pi_i(\mathbf{X})\big) \Big] \Big\langle \nabla_{\mathfrak{s}} f\big(\mathbf{X}\big), \frac{\partial(\mathbf{X})}{\partial \lambda} \Big\rangle J_i d\boldsymbol{u} && (21) \\
&+ \int_{\partial U} \Big[I_i\big(\pi_i(\mathbf{X})\big) - \big(w_i f\big(\mathbf{X}\big) + \gamma_i\big) \Big]^2 \Big\langle Q \frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial s}, \frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial \lambda} \Big\rangle ds && (22) \\
&+ W \sum_{j=1(j\neq i)}^{N_c} (\|\Omega\| - d_{ij}^o) \frac{\partial \|\Omega\|}{\partial \lambda} \Bigg\}^{\mathsf{T}}, && (23)
\end{aligned}$$

in which Δt is a well-chosen positive scalar (discussed in the next section). In addition,

$$\frac{\partial \|\Omega\|}{\partial \lambda} = \begin{cases} 0 & \text{if } \lambda = \big(L_x, L_y\big)^{\mathsf{T}} \text{or } \big(\zeta_0, \eta_0\big)^{\mathsf{T}} \\ \frac{\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j}{\|\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j\|} & \text{if } \lambda = \mathbf{t}_i \\ \frac{\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j}{\|\mathbf{t}_i - \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} \mathbf{t}_j\|} \mathsf{R}_i \mathsf{R}_j^{\mathsf{T}} [\mathbf{t}_j]_{\mathbf{x}} \mathsf{R}_j \Psi & \text{if } \lambda = \vec{\omega}_i \end{cases}. \quad (24)$$

All notations have already been introduced in Table 2.

3.4 Inexact line search

The gradient descent method states that compared to any other directions, error function $E_{calib}(\lambda)$ descends the most when λ is added by a vector $\Delta\lambda$ whose direction is along $-\nabla_\lambda E_{calib}$. The magnitude of $\Delta\lambda$ (denoted by $\|\Delta\lambda\|$) determines the computation efficiency: On the one hand, the gradient descent method with a large

$\|\Delta\lambda\|$ is less prone to determine a convergent value satisfying $\frac{\partial E_{calib}}{\partial \lambda} = 0$ than a small $\|\Delta\lambda\|$; on the other hand, the gradient descent method with a small $\|\Delta\lambda\|$ requires more repetitive evaluations of $E_{calib}(\lambda)$ and $\nabla_\lambda E_{calib}$, which wastes computational resources, than a large $\|\Delta\lambda\|$. Hence, an optimal $\|\Delta\lambda\|$ is necessary for satisfying both stability and computation issues. However, attempting to obtain an optimal $\|\Delta\lambda\|$ complicates the entire problem while the gradient descent method is used to locate a local minimum of $E_{calib}(\lambda)$: Since we use the gradient descent method to resolve one optimization problem (locating a local minimum of $E_{calib}(\lambda)$), why do we bother to introduce another optimization problem (optimizing $\|\Delta\lambda\|$) while the original one is not yet solved?

Several numerical methods were explored to find an appropriate $\|\Delta\lambda\|$. Most of these methods rely on a reasonable estimation of region in which a local minimum resides. By trial and error, we learned from previous 3-D reconstruction experiments that any camera parameter element with a relative error larger than 4% of the original values (camera parameters obtained through camera calibration tools) would entirely fail the reconstruction result produced by the VWASS. Therefore, the region for us to locate a local minimum around the current parameter estimation (denoted by λ_0) is confined within the line segment determined by λ_0 and $(\lambda_0 + \alpha_s \Delta\lambda)$, in which α_s is a positive scalar and is limited by

$$\left| max\Big((\alpha_s \Delta\lambda)./(\lambda_0)\Big) \right| < 4\%. \tag{25}$$

In this equation, "./" represents the element-wise division.

Based on the determined range, the golden section search and the backtracking line search (the Armijo rule) were used to locate local minima along the gradient descent direction. However, considering convergence rates, stability, and computation complexity, the strong Wolfe conditions [18, 38] (explained in appendix B) are eventually adopted to determine an appropriate $\|\Delta\lambda\|$.

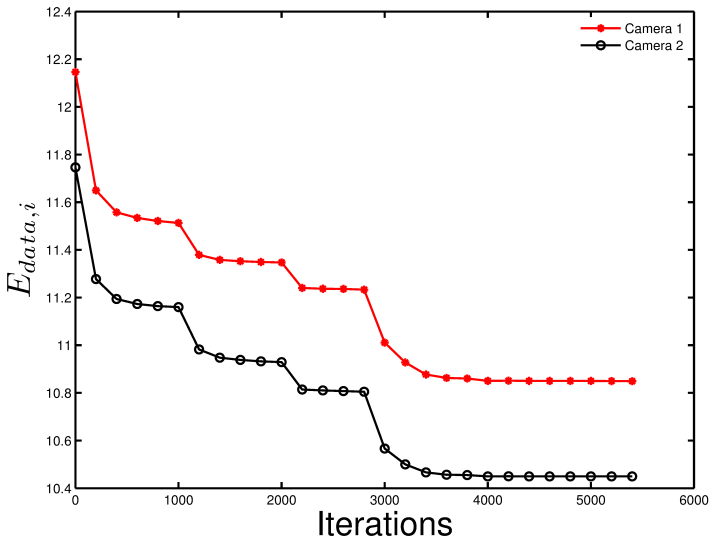


Figure 6: The changes of $E_{data,i}$ with respect to the sequential refinement of each type of camera parameters. $\|\mathbf{t}_i\|$ first, then $\vec{\omega}_i$, $\left(\zeta_0, \eta_o\right)^\mathsf{T}$, and $\left(L_x, L_y\right)^\mathsf{T}$.

3.5 The order of refining camera parameters

In most practical cases, we do not know which camera parameters are inaccurately estimated through camera calibration tool [6] or [7]. To guarantee that all deviated camera parameters are refined, we sequentially execute the calibration refinement process for each type of camera parameter. The heuristically-determined execution order is as follows: translation vector $\|\mathbf{t}_i\|$, rotation vector $\vec{\omega}_i$, principal point position $\left(\zeta_0, \eta_o\right)^\mathsf{T}$, and focal length $\left(L_x, L_y\right)^\mathsf{T}$. Figure 6 illustrates how $E_{data,i}$ changes with respect to the refinement of each type of camera parameters based on the aforementioned refinement order.

However, the sequential execution of the camera parameter refinement wastes computational resources on the repetitive calculation of terms that are non-relevant to camera parameters λ, such as $\int_U \left[I_i\left(\pi_i(\mathbf{X})\right) - \left(w_i f\left(\mathbf{X}\right) + \gamma_i\right)\right]^2 J_i d\boldsymbol{u}$ in Term (22). Seemingly, treating all camera parameters as components of a long vector, Λ,[4] and attaining a local minimum of E_{calib} along the corresponding gradient descent direction, $-\nabla_\Lambda E_{calib}$, is a means of saving computational efforts. Nevertheless, not all the

[4]In other words, Λ is defined as $\Lambda = [\mathbf{t}_i^\mathsf{T}, \vec{\omega}_i^\mathsf{T}, \left(L_x, L_y\right)^\mathsf{T}, \left(\zeta_0, \eta_o\right)^\mathsf{T}]^\mathsf{T}$.

camera parameters belong to the same Euclidean space, so the resulting gradient vector,

$$\nabla_{\Lambda} E_{calib} = \left[\begin{array}{cccc} (\nabla_{\mathbf{t}_i^{\mathsf{T}}} E_{calib})^{\mathsf{T}} & (\nabla_{\vec{\omega}_i^{\mathsf{T}}} E_{calib})^{\mathsf{T}} & (\nabla_{\left(L_x, L_y\right)^{\mathsf{T}}} E_{calib})^{\mathsf{T}} & (\nabla_{\left(\zeta_0, \eta_o\right)^{\mathsf{T}}} E_{calib})^{\mathsf{T}} \end{array} \right]^{\mathsf{T}}, \tag{26}$$

is physically meaningless and numerically useless.

Inspired by [24], we can weigh the components in Λ with appropriate scalars to produce resulting components with similar magnitudes. As a result, no particular component in the resulting vector (denoted by Λ') is numerically dominant during the usage of $\nabla_{\Lambda'} E_{calib}$. This strategy was used in [21, p.57] to improve the numerical condition of computing the induced homography. In the following context, we use P_i to denote the forward projection matrix of the i^{th} camera (that is, $P_i = \mathrm{K}_i \left[\mathsf{R}_i \mid \mathbf{t}_i\right]$[5]) and define a normalized forward projection matrix, ${}_nP_i$, in which

$${}_nP_i = TP_iU.$$

In this equation, T and U are two similarity transformation matrices whose dimensions are three-by-three and four-by-four, respectively. Suggested by [21, p.57], U is chosen as

$$U = \left[\begin{array}{cc} \mathbf{I}_{3\times 3} & 0 \\ 0 & s_u^{-1} \end{array} \right], \tag{27}$$

in which $s_u = \frac{\sum_{i=1}^{N_c} \|\mathbf{t}_i\|}{N_c}$ and T is

$$T = \left[\begin{array}{cc} s_T \mathbf{I}_{2\times 2} & s_T \overbrace{\left[\begin{array}{c} \frac{-w}{2} \\ \frac{-h}{2} \end{array} \right]}^{\mathbf{t_T}} \\ 0 & 1 \end{array} \right] = \left[\begin{array}{ccc} \frac{2}{w+h} & 0 & -\frac{w}{w+h} \\ 0 & \frac{2}{w+h} & -\frac{h}{w+h} \\ 0 & 0 & 1 \end{array} \right],$$

[5] P_i and π_i are essentially the same mathematical operation. The only difference between P_i and π_i is $\pi_i(\mathbf{X}) = (P_i\mathbf{X})_n$, in which the subscript, n, indicates the operation that converts the homogeneous coordinates in the parentheses to the Cartesian coordinates.

where $s_T = \frac{2}{w+h}$ and w and h are the width and the height of the i^{th} image, respectively. Note that we assume all the images have the same width and height so T is universal to all normalized forward projection matrices.

The reason why ${}_nP_i$ is such defined is explained as follows: ${}_nP_i$ is designed to project normalized 3-D points of $\mathbf{X}$, denoted by ${}_n\mathbf{X}$, to normalized 2-D points of $\hat{\mathbf{x}}_e$[6], denoted by ${}_n\hat{\mathbf{x}}_e$. ${}_n\mathbf{X}$ is defined as ${}_n\mathbf{X} = U^{-1}\mathbf{X}$ and ${}_n\hat{\mathbf{x}}_e$ is defined as ${}_n\hat{\mathbf{x}}_e = {}_nP_i\ {}_n\mathbf{X}$. With this linear transformation, each element in ${}_n\mathbf{X}$ or ${}_n\hat{\mathbf{x}}_e$ has a numerical value whose magnitude is approximately 1. In addition, ${}_nP_i$ is practically a new forward projection whose all camera parameters have magnitudes approximately 1, which is demonstrated as follows:

$$
\begin{aligned}
{}_nP_i &= TP_iU \\
&= \begin{bmatrix} s_T\mathbf{I}_{2\times 2} & \mathbf{t_T} \\ 0 & 1 \end{bmatrix} \underbrace{\mathrm{K}_i \begin{bmatrix} \mathsf{R}_i & | & \mathbf{t}_i \end{bmatrix}}_{P_i} \begin{bmatrix} \mathbf{I}_{3\times 3} & 0 \\ 0 & s_u^{-1} \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} s_T\mathbf{I}_{2\times 2} & \mathbf{t_T} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} L_x & 0 & \zeta_0 \\ 0 & L_y & \eta_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{K}'_i} \underbrace{\begin{bmatrix} \mathsf{R}_i & | & \mathbf{t}_i \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3\times 3} & 0 \\ 0 & s_u^{-1} \end{bmatrix}}_{\begin{bmatrix} \mathsf{R}_i & | & \mathbf{t}'_i \end{bmatrix}} \\
&= \underbrace{\begin{bmatrix} s_T \begin{bmatrix} L_x & 0 \\ 0 & L_y \end{bmatrix} & s_T \begin{bmatrix} \zeta_0 \\ \eta_0 \end{bmatrix} + \mathbf{t_T} \\ 0 & 1 \end{bmatrix}}_{\mathrm{K}'_i} \underbrace{\begin{bmatrix} \mathsf{R}_i & | & s_u^{-1}\mathbf{t}_i \end{bmatrix}}_{\begin{bmatrix} \mathsf{R}_i & | & \mathbf{t}'_i \end{bmatrix}}.
\end{aligned} \tag{28}
$$

One can verify that the extrinsic and intrinsic parameters in equation (28) have magnitude around 1.

The definition of E_{calib} in equation (18) requires modifications to reflect the

[6] $\hat{\mathbf{x}}_e$ is the homogeneous coordinate of $\hat{\mathbf{x}}$. Likewise, ${}_n\hat{\mathbf{x}}_e$ is the homogeneous coordinate of ${}_n\hat{\mathbf{x}}$, which is used in the next page.

changes of the forward projection. As such, we define a modified cost function of E_{calib}, denoted by ${}_nE_{calib}$, with ${}_nP_i$, ${}_n\mathbf{X}$, and ${}_n\hat{\mathbf{x}}_e$ introduced to substitute for their counterparts in equation (18), yielding

$${}_nE_{calib} = \sum_{i=1}^{N_c} \int_{{}_nR_i} \Big[I_i\big({}_n\hat{\mathbf{x}}\big) - \big(w_i f\big({}_n\pi_i^{-1}({}_n\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\ {}_n\hat{\mathbf{x}} + \frac{W}{2} \sum_{i=1}^{N_c-1} \sum_{j=i+1}^{N_c} (\|\mathsf{R}_i^{\mathsf{T}}\mathbf{t}_i' - \mathsf{R}_j^{\mathsf{T}}\mathbf{t}_i'\| - d_{ij}'^o)^2. \tag{29}$$

In this equation, all symbols with left subscript n represent the normalized version of their counterparts in equation (18).

The analytical form of the derivatives of ${}_nE_{calib}$ with respect to the normalized camera parameters can be obtained through the change of variables and the results in Tables 1 and 2. If we denote the first and the second terms in equation (18) ${}^1E_{calib}$ and ${}^2E_{calib}$, respectively, and the first and the second term in equation (29) ${}^1_nE_{calib}$ and ${}^2_nE_{calib}$, respectively, then ${}^1E_{calib} = s_T^{-2}\,({}^1_nE_{calib})$ given that

$$\begin{aligned}
{}^1E_{calib} &= \sum_{i=1}^{N_c} \int_{R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}} \\
&= \sum_{i=1}^{N_c} \int_{{}_nR_i} \Big[I_i\big(\frac{{}_n\hat{\mathbf{x}} - \mathbf{t_T}}{s_T}\big) - \big(w_i f\big(\pi_i^{-1}(\frac{{}_n\hat{\mathbf{x}} - \mathbf{t_T}}{s_T})\big) + \gamma_i\big) \Big]^2 s_T^{-2} d\ {}_n\hat{\mathbf{x}} \\
&= s_T^{-2}\,({}^1_nE_{calib}).
\end{aligned}$$

In addition, ${}^2E_{calib} = s_u^2\,({}^2_nE_{calib})$.

Therefore, after taking the derivatives of ${}_nE_{calib}$ with respect to the normalized intrinsic camera parameters in equation (28), we obtain

$$\begin{cases}
\nabla_{(L_x',L_y')^{\mathsf{T}}}({}_nE_{calib}) = \nabla_{(L_x',L_y')^{\mathsf{T}}}({}^1_nE_{calib}) = s_T \nabla_{(L_x,L_y)^{\mathsf{T}}}({}^1E_{calib}) \\
\nabla_{(\zeta_0',\eta_0')^{\mathsf{T}}}({}_nE_{calib}) = \nabla_{(\zeta_0',\eta_0')^{\mathsf{T}}}({}^1_nE_{calib}) = s_T \nabla_{(\zeta_0,\eta_0)^{\mathsf{T}}}({}^1E_{calib})
\end{cases}. \tag{30}$$

As for the derivative of ${}_nE_{calib}$ with respect to translation vector $\mathbf{t}'_i$, we have

$$\begin{aligned}
\frac{\partial({}_nE_{calib})}{\partial \mathbf{t}'_i} &= \frac{\partial({}^1_nE_{calib})}{\partial \mathbf{t}'_i} + \frac{W}{2}\frac{\partial({}^2_nE_{calib})}{\partial \mathbf{t}'_i} \\
&= s_T^2\frac{\partial({}^1E_{calib})}{\partial \mathbf{t}'_i} + \frac{W}{2}s_u^{-2}\frac{\partial({}^2E_{calib})}{\partial \mathbf{t}'_i} \\
&= s_T^2\frac{\partial({}^1E_{calib})}{\partial \mathbf{t}_i}\frac{\partial \mathbf{t}_i}{\partial \mathbf{t}'_i} + \frac{W}{2}s_u^{-2}\frac{\partial({}^2E_{calib})}{\partial \mathbf{t}_i}\frac{\partial \mathbf{t}_i}{\partial \mathbf{t}'_i} \\
&= s_T^2 s_u\frac{\partial({}^1E_{calib})}{\partial \mathbf{t}_i} + \frac{W}{2}s_u^{-1}\frac{\partial({}^2E_{calib})}{\partial \mathbf{t}_i}.
\end{aligned}$$

Hence,

$$\nabla_{(\mathbf{t}'_i)^\top}({}_nE_{calib}) = s_T^2 s_u \nabla_{(\mathbf{t}_i)^\top}({}^1E_{calib}) + \frac{W}{2}s_u^{-1}\nabla_{(\mathbf{t}_i)^\top}({}^2E_{calib}). \tag{31}$$

Likewise,

$$\nabla_{(\vec{\omega}_i)^\top}({}_nE_{calib}) = s_T^2 \nabla_{(\vec{\omega}_i)^\top}({}^1E_{calib}) + \frac{W}{2}s_u^{-2}\nabla_{(\vec{\omega}_i)^\top}({}^2E_{calib}). \tag{32}$$

Equations (30)–(32) indicate that we can simply multiply the gradient vectors in Tables 1 and 2 with the parameters introduced in T and U, which are used to normalize P_i to produce the components in $\nabla_{(\Lambda')}({}_nE_{calib})$.

3.6 Validation

We conducted an experiment on two images taken at the same time moment from an offshore platform at the Crimean Peninsula, the Black Sea. The region of interest, marked with white in Figure 3.6, is a square with a side length of 12.8 meters. Before cameras are used, they are manually calibrated to the optimal extent with the Camera Calibration Toolbox for MATLAB [6]. With these camera parameters, we first reconstructed the ocean waves, shown in Figure 9, using the variational 3-D reconstruction algorithm introduced in section 2.3.1. After generating a 3-D model, we alternately performed the calibration refinement process and the original 3-D reconstruction algorithm, yielding the convergent camera parameters and the convergent reconstruction results shown in Figures 10 (the elevation map) and 12(b) (the radiance map).

From the perspective of optimization, alternately performing the calibration refinement and the 3-D reconstruction algorithms is a minimization process of E with respect to λ (the camera parameters) and $\bar{\lambda}$ (Z and f in error functional (5)). This process, based on the assumption that the initial estimation of λ is "accurate enough," is explained in Figure 8: At the very beginning, we attain a local minimum of E at $\bar{\lambda}_1$ through iteratively solving PDEs (7)–(10). In this step, λ (the camera parameters, determining the coefficients of the PDEs) are fixed. Then, by fixing the current $\bar{\lambda}$, we attain a local minimum of E at position λ_2 by iteratively solving ODEs (21)–(23). $E(\lambda_{1'}, \bar{\lambda}_1)$ is originally a local minimum with respect to $\bar{\lambda}$, however, with the change of the first argument from $\lambda_{1'}$ to λ_2, $E(\lambda_2, \bar{\lambda}_1)$ is no longer a local minimum with respect to its second argument (it changes from the green dashed-curve to the red one in the left figure), so we evolve $\bar{\lambda}$ to diminish E, attaining a local minimum at position $\bar{\lambda}_3$. On the condition that the initial estimation of λ is not far away from the true value, we can get the optimal estimations of λ and $\bar{\lambda}$ by repeating the aforementioned procedure.

The greatest advantage of this strategy is that the optimization process with a relatively low convergence rate (the calibration refinement process, because of the usage of the gradient descent method) can start with a "good enough" initial estimation generated by the other process with a relatively high convergence rate (the 3-D reconstruction of Z and f, because of the usage of the the multigrid method [8, 21]), which can efficiently accelerate both optimization processes.

We applied our calibration refinement program to a convergent reconstruction of ocean waves and showed the resulting elevation map in Figure 10. The reconstructed elevation map without the calibration refinement procedure is shown in Figure 9 for comparison, and the difference between Figure 9 and Figure 10 is presented in Figure 11. One can see that the algorithm in section 2.3.1 can generate a promising ocean wave model solely based on the camera parameters obtained through [6]. However,

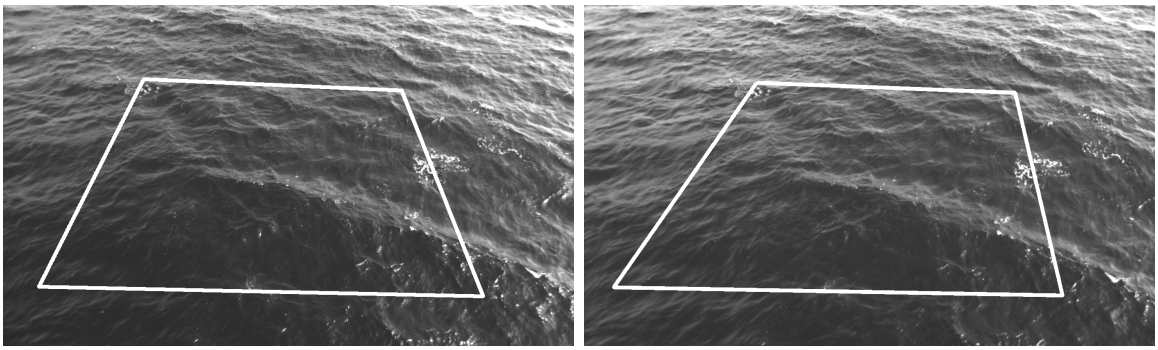


Figure 7: Input images (the white quadrilateral is the region of interest). Because the images were taken by different cameras, the brightness levels of both images are different. Hence, two coefficients, w_i and γ_i, are introduced in E_{data} in equation (6) to adjust the brightness level of each image, causing the adjusted brightness to better approximate the reprojection of f than the original brightness.

with the camera parameters refined by our algorithm, the new reconstruction model exhibits more delicate details than that without the refinement procedure: The left half of the reconstruction shows significant height changes. These height changes rectify the skewness of the elevation map from 0.611 to 0.265, which agrees with the theoretical skewness range, 0 – 0.3.

The reconstructed radiance maps before and after the calibration process as well as their difference are shown in Figures 3.6 and 13, respectively. Although the variances on the radiance maps are not clear on Figure 3.6, one can still locate the changes and observe their magnitudes from Figure 13. The occurrences of the radiance changes mostly agree with those of the height changes on the elevation maps: Most distinct changes occur on the left half of the reconstructed region, and the changes at the bottom right regions of the elevation and radiance maps are subtle.

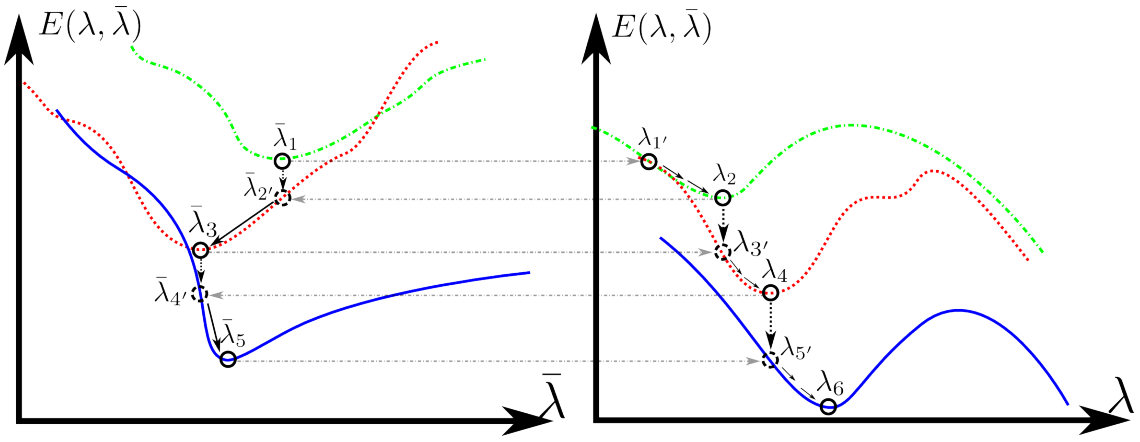


Figure 8: The minimization process of E with respect to λ (the camera parameters) and $\bar{\lambda}$ (Z and f in error functional (5)). The strategy is to reach a local minimum of E based on an "accurate enough" initial estimation of λ through alternate minimizations of E with respect to λ and $\bar{\lambda}$.

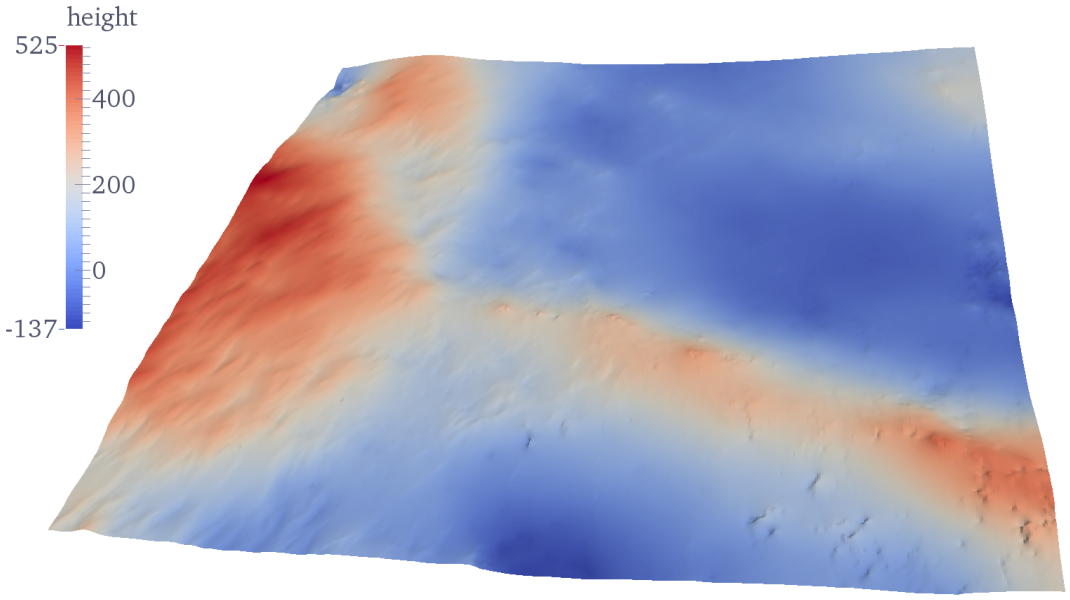


Figure 9: The reconstructed elevation map, determined as one of the minimizers of error functional (5), takes the image pair in Figure 3.6 as inputs.

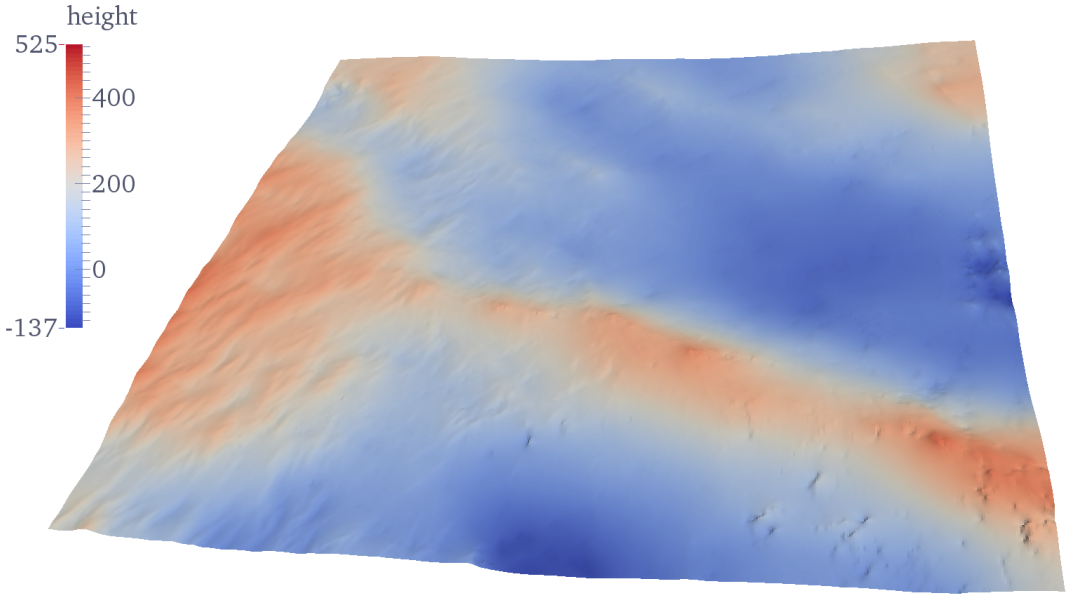


Figure 10: The elevation map generated through the joint operations of the calibration refinement and the 3-D reconstruction algorithms. This experiment demonstrates that our calibration refinement algorithm helps the 3-D reconstruction algorithm capture more delicate details and improves the skewness of the reconstructed wave model.

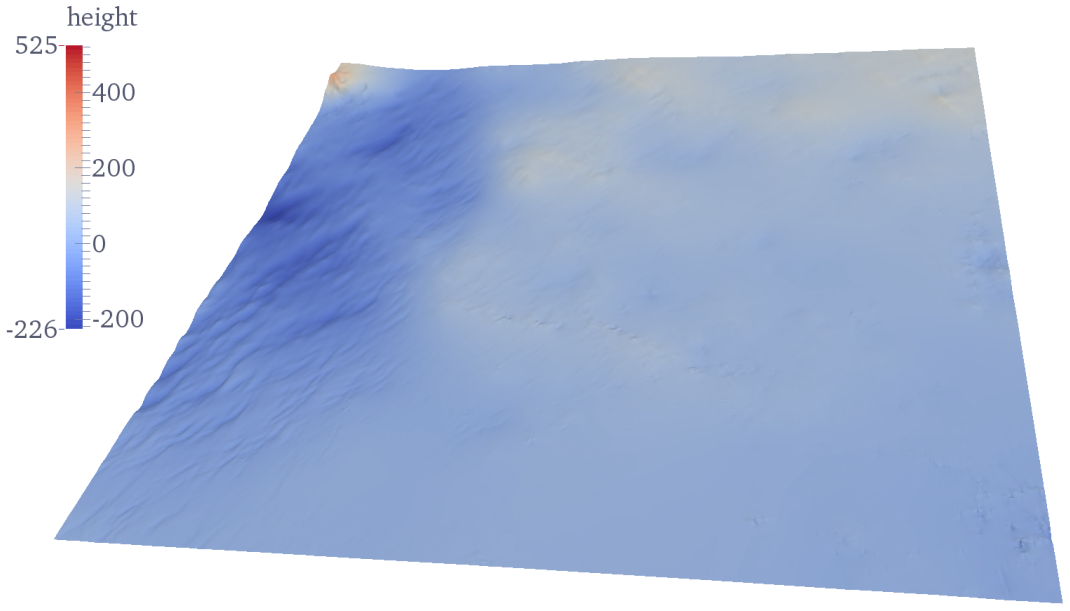
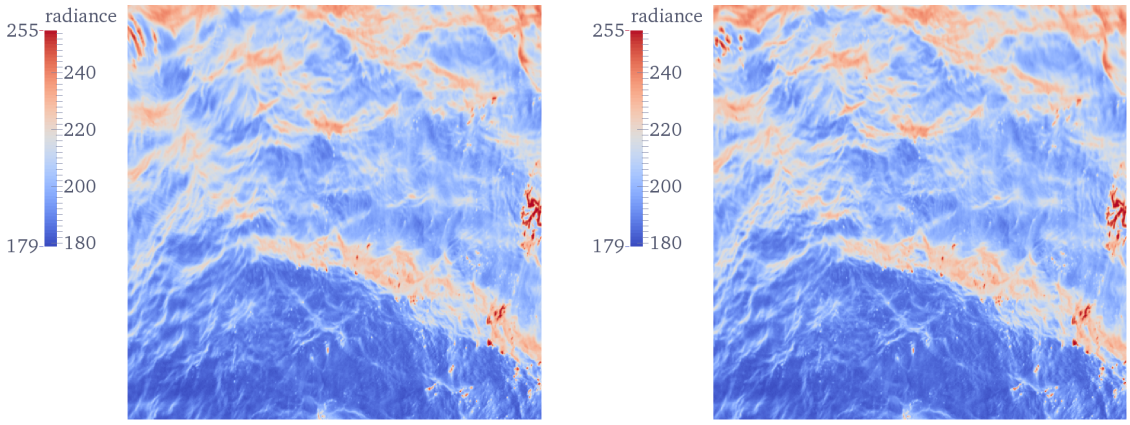


Figure 11: Difference map between Figure 9 and Figure 10. Significant changes appear locally on the left half.



(a) (b)

Figure 12: This figure contains two radiance maps. The left one is the result of the variational 3-D reconstruction; the right one is obtained through the joint operations of the calibration refinement and the variational 3-D reconstruction algorithms. Note that some features on the upper left corner of the left figure look more blurred than the corresponding parts on the other figure.

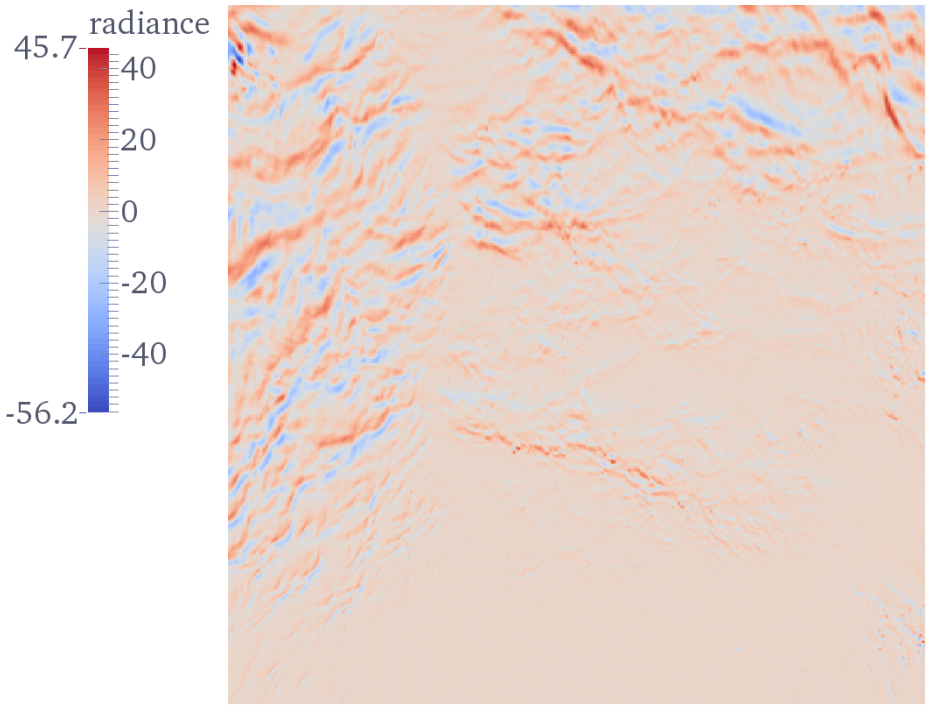


Figure 13: The difference between the two radiance maps shown in Figure 3.6. Similar to Figure 11, most discrepancies on the radiance maps occur on the left half of the region of interest. Note that the hue bar in this figure has a different scale from the ones in Figure 3.6.

CHAPTER IV

SYNTHETIC DATA FOR VALIDATION: SYNCHRONIZED SNAPSHOTS AND VIDEOS

Validating the correctness of the gradient descent vectors listed in Table 1 using real datasets is difficult given that cameras are manually calibrated to the optimal extent before usage. Under this condition, some of the camera parameter deviations in real datasets are too subtle to be used to validate of the corresponding gradient descent vectors. Even though we can intentionally miscalibrate cameras, we are unable to control or quantify the artificial errors introduced to each type of camera parameter. Hence, to validate the joint operations of the calibration refinement and the variational 3-D reconstruction algorithms, we establish synthetic datasets using computer graphics techniques in this chapter.

4.1 Computer graphics versus 3-D reconstruction

Computer graphics applications work as the reverse operations of 3-D reconstruction methods. As introduced in chapter 1, 3-D reconstruction methods take 2-D measurements and the information concerning the observation positions and orientations as inputs, recovering the 3-D coordinates of the observed 2-D features as outputs. Computer graphics applications operate in the opposite way: imaginary objects are created in virtual reality and viewed from different configured observation positions. Inspired by the relationship between computer graphics and 3-D reconstruction, we use computer graphics tools, such as OpenGL, to generate the inputs required by our 3-D reconstruction algorithm. As a result, we can judge the performance of our algorithms by simply comparing their inputs and outputs.

To that end, we use OpenGL to create a region of synthetic ocean waves in virtual reality and set up multiple imaginary cameras to "take pictures" of the synthetic ocean surface. The synthetic ocean surface will be a dense 3-D point cloud in which the coordinates of all points are known (because we create them), and a pattern will be rendered on the dense point cloud as the radiance map of the synthetic waves. The pictures taken by imaginary cameras from different angles are simulated by the contents rendered in the application programming interface (API) windows, which can be fetched from the memory of the computer graphics card.

The operations mentioned above can be easily handled with a simple understanding of OpenGL. However, additional information is required by 3-D reconstruction algorithms: the extrinsic and intrinsic parameters of the imaginary cameras. Because the conventional coordinate setups of most 3-D reconstruction algorithms follow the configuration in [6], which are different from those of virtual reality created by OpenGL, we need to derive the formulas that convert the coordinate systems used by OpenGL and 3-D reconstruction applications.

With all this aforementioned information available, we can 1) create a patch of ocean surface in virtual reality and fetch their renderings, 2) obtain corrupted camera parameters by adding artificial errors to the true ones converted from the environment variables of virtual reality, and 3) jointly perform the calibration refinement algorithm and 3-D reconstruction algorithm to see how the reconstruction is rectified. Note that this approach is particularly suitable for variational 3-D reconstruction methods given that their inputs and outputs are both dense point clouds. Hence, by deliberately discretizing the elevation and radiance maps of the output with the same number of samples as the input point cloud, we can verify the accuracy of the reconstructed model by differentiating it with the input point cloud and the selected radiance.

The purpose of this chapter is to demonstrate the derivation of the formulas that transform the environment variables of virtual reality into conventional coordinate

arrangements used in 3-D reconstruction applications. Although generating a patch of synthetic ocean surface and rendering it into images or videos require advanced knowledge of OpenGL, we do not particularly emphasize the technical details since they can be easily found in most OpenGL books.

4.2 Coordinate system configuration of OpenGL

As introduced in section 2.2, the perspective projection that most stereo computer vision algorithms are based on is the pin-hole projection, which is demonstrated in Figures 2 and 14. Although the images captured by modern cameras are usually accompanied by lens distortions, these distortions can be removed through some mathematical algorithms. Hence, we do not consider the effects of lens in the context of this thesis. All parameters necessary for describing the perspective projection have been denoted and explained in section 2.2.

The perspective transformation of OpenGL is illustrated in Figure 15. In practice, the perspective transformation is not the only projection method provided by OpenGL, but we are only interested in it because this transformation corresponds with the pin-hole projection concept used in most stereo computer vision applications. OpenGL [1, 9, 43] uses two coordinate systems—$\left(x_o, y_o, z_o\right)^\mathsf{T}$ and $\left(x'_o, y'_o, z'_o\right)^\mathsf{T}$—to locate 3-D points in the world coordinate system of virtual reality. The relationship between the two coordinate systems is $[x'_o, y'_o, z'_o, w'_o]^\mathsf{T} = M_{model}\,[x_o, y_o, z_o, w_o]^\mathsf{T}$, in which $[x_o, y_o, z_o, w_o]^\mathsf{T}$ and $[x'_o, y'_o, z'_o, w'_o]^\mathsf{T}$ are the homogeneous coordinates of $[x_o, y_o, z_o]^\mathsf{T}$ and $[x'_o, y'_o, z'_o]^\mathsf{T}$, respectively. M_{model} is a four-by-four matrix used to facilitate the creation of objects at various poses. Imagine that a designer wants to use OpenGL to create a model of a gull diving into the surface of a lake. Instead of directly generating a gull diving into water at a certain angle, the designer can generate a flying gull at the origin of virtual reality (perhaps by importing the points obtained from a laser scanner) and then use M_{model} to adjust its position and orientation.

The coordinate system of the observer in OpenGL is usually called the eye coordinate system—$[x_e, y_e, z_e]^\mathsf{T}$. The transformation from $[x'_o, y'_o, z'_o, w'_o]^\mathsf{T}$ to $[x_e, y_e, z_e, w_e]^\mathsf{T}$ is linear, defined as $[x_e, y_e, z_e, w_e]^\mathsf{T} = M_{view}\,[x'_o, y'_o, z'_o, w'_o]^\mathsf{T}$. Therefore,

$$\begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix} = M_{modelview} \begin{bmatrix} x_o \\ y_o \\ z_o \\ w_o \end{bmatrix},$$

where $M_{modelview} = M_{view} M_{model}$. OpenGL does not provide any built-in APIs for users to separately access M_{model} and M_{view}; only their product, $M_{modelview}$, can be accessed through a certain number of APIs.

OpenGL allows users to cull some created points in virtual reality from being rendered since rendering is a computationally expensive task. Only the points enclosed within the rectangular view frustum (shown in Figure 15 or 16(a)) are rendered for visualization; other points are simply ignored. The view frustum is determined by six parameters : l, r, t, b, f, and n, as marked in the figures. The first four parameters can be positive or negative numbers, but f and n are restricted to positive ones given the coordinate configuration specified in Figure 15 or 16(a), in which the origin of the eye coordinate system is set at the vertex of the view frustum and the positive direction of z_e is configured to point away from the bottom of the frustum.

The coordinate readings in the eye coordinate system ($[x_e, y_e, z_e, w_e]^\mathsf{T}$) are subsequently converted to another coordinate system named the clip coordinates. Denoted by $[x_c, y_c, z_c, w_c]^\mathsf{T}$, the clip coordinates are obtained through $[x_c, y_c, z_c, w_c]^\mathsf{T} = M_{proj}\,[x_e, y_e, z_e, w_e]^\mathsf{T}$, in which M_{proj} is a four-by-four matrix whose function is similar to the forward projection in equation (2), π_i, except that $[x_c, y_c, z_c, w_c]^\mathsf{T}$ does not represent the coordinates defined on sensor planes. Instead, $[x_c, y_c, z_c, w_c]^\mathsf{T}$ is a set

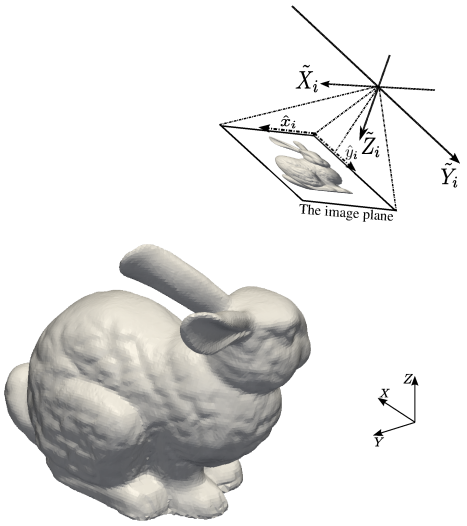


Figure 14: This figure, as a comparison to Figure 15, represents the concepts of the coordinate transformations adopted by most stereo computer vision applications. The symbols shown in this figure have been introduced in section 2.2. Note that the lens distortion effects on projection are not considered because they can be mathematically removed.

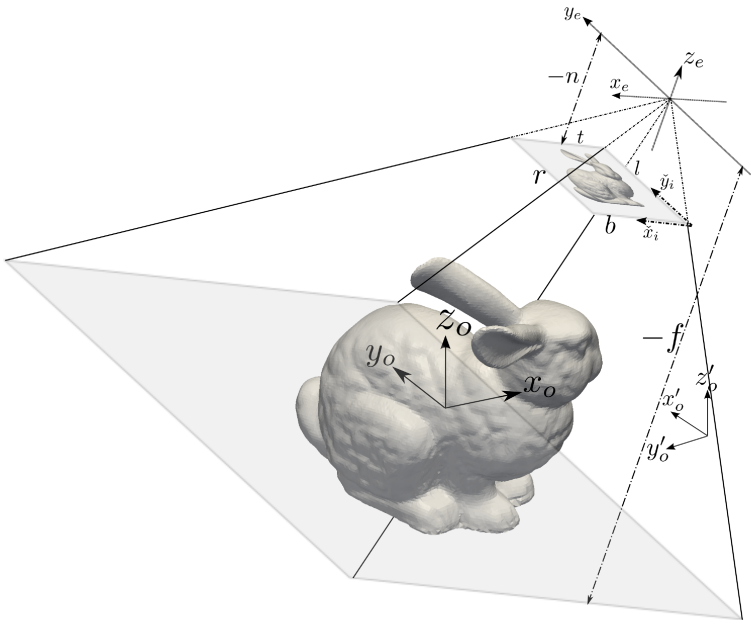


Figure 15: The perspective projection of OpenGL. Only the points inside the frustum will be rendered for users to visualize. Compared to Figure 14, this figure has additional parameters l, r, t, b, n, and f for constructing the frustum. In addition, two coordinate systems—$\left(x_o, y_o, z_o\right)^{\mathsf{T}}$ and $\left(x'_o, y'_o, z'_o\right)^{\mathsf{T}}$—are offered to facilitate the creation of 3-D objects.

of intermediate coordinates whose normalization ($[x_c/w_c, y_c/w_c, z_c/w_c]^\mathsf{T}$,[1] denoted by $[x_n, y_n, z_n]^\mathsf{T}$) is scaled to the range of $[-1, 1]$. In OpenGL, $[x_n, y_n, z_n]^\mathsf{T}$ is called the normalized device coordinates (NDC, shown in Figure 16(b)); their upper bounds and lower bounds are 1 and -1, respectively. The boundaries of the view frustum are mapped to the bounds of NDC through M_{proj}: l, b, and $-n$ are mapped to -1; r, t, and $-f$ are mapped to 1. Since $[x_n, y_n, z_n]^\mathsf{T}$ represents the conditions of any visible point within the frustum, elements of $[x_n, y_n, z_n]^\mathsf{T}$ can only range between -1 and 1.

The multiplications of x_n and y_n with the half width and half height of the viewport, respectively, determine the projection of point $\mathbf{P}$ in the eye coordinate system onto the viewport, as shown in Figure 16(c). Note that although w_c can be any nonzero value, OpenGL chooses to set $w_c = -z_e$. As such, the depth information of a point is preserved after projection.

Based on the aforementioned description, we can use the concept of similar triangles to obtain the relationship between x_n and x_e, yielding $x_n = \frac{l+r}{l-r} + \frac{2n}{l-r}\frac{x_e}{z_e}$. Likewise, y_n and y_e are related by $y_n = \frac{b+t}{b-t} + \frac{2n}{b-t}\frac{y_e}{z_e}$. By plugging the two equations into M_{proj}, we have

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = M_{proj} \begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix} = \begin{bmatrix} \frac{2n}{(r-l)} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & m_{33} & m_{34} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix}. \tag{33}$$

The remaining unknowns, m_{33} and m_{34}, can be solved using the the fact that the boundaries of the view frustum are mapped to the boundaries of the NDC system. Therefore, $-f$ and $-n$ in the eye coordinate system are mapped to $z_n = 1$ and

[1] Also called the perspective division in [43].

$z_n = -1$ in NDC, respectively. As a result, $m_{33} = \frac{-(f+n)}{f-n}$ and $m_{34} = \frac{-2fn}{f-n}$. Therefore,

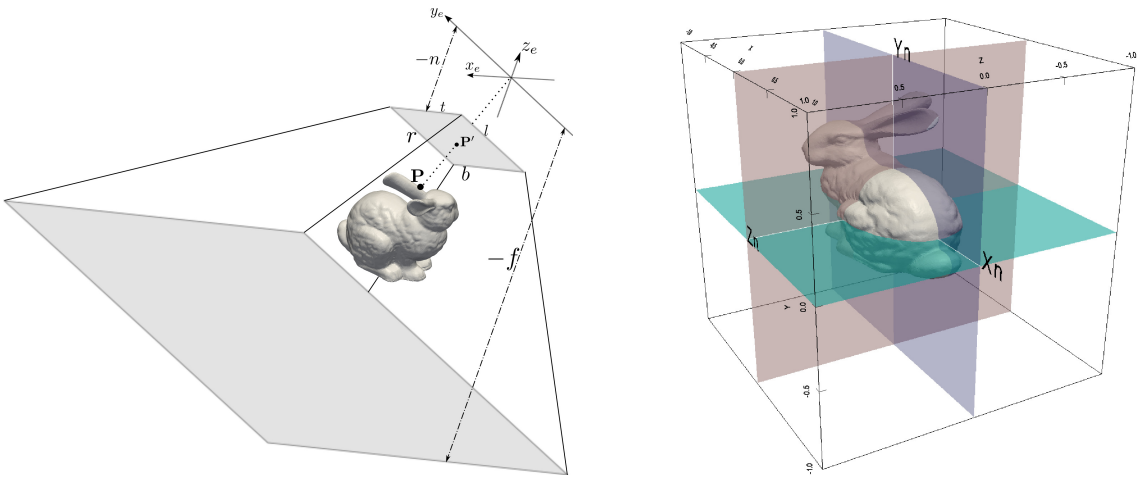
$$M_{proj} = \begin{bmatrix} \frac{2n}{(r-l)} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}. \tag{34}$$

Note that the transformations from x_e and y_e to x_n and y_n, respectively, are linear, whereas the transformation from z_e to z_n is nonlinear because of the effect of normalization (perspective division). Finally, we would like to represent l, r, t, b, f, and n in terms of the elements of M_{proj} since these elements can be fetched from OpenGL APIs, so we obtain

$$\begin{cases} r = & \frac{(M_{proj})_{34}\left((M_{proj})_{13}+1\right)}{(M_{proj})_{11}\left((M_{proj})_{33}-1\right)} \\ l = & \frac{(M_{proj})_{34}\left((M_{proj})_{13}-1\right)}{(M_{proj})_{11}\left((M_{proj})_{33}-1\right)} \\ t = & \frac{(M_{proj})_{34}\left((M_{proj})_{23}+1\right)}{(M_{proj})_{22}\left((M_{proj})_{33}-1\right)} \\ b = & \frac{(M_{proj})_{34}\left((M_{proj})_{23}-1\right)}{(M_{proj})_{22}\left((M_{proj})_{33}-1\right)} \\ n = & \frac{(M_{proj})_{34}}{(M_{proj})_{33}-1} \\ f = & \frac{(M_{proj})_{34}}{(M_{proj})_{33}+1} \end{cases}. \tag{35}$$

4.3 Coordinate conversions from OpenGL to stereo computer vision

By comparing both the coordinate system arrangements of 3-D reconstruction (in section 2.2) and OpenGL (in section 4.2), we conclude their notation correspondences in Table 3. The purpose of this section is to derive the formulas yielding R_i, $\mathbf{t}_i$, L_x, L_y, ζ_0, and η_0—the extrinsic and intrinsic camera parameters—from the OpenGL variables listed in the second row: we use an OpenGL API called "glGetFloatv" to



(a) View frustum

(b) Normalized device coordinates (NDC)

(c) The rendering of points in the view frustum

Figure 16: Figure 16(a) shows a view frustum enclosing a bunny. Only the points located within the frustum are eventually rendered on the viewport. The frustum is determined when 1) parameters l, r, t, b, f, and n are specified and 2) the origins and orientations of the eye coordinate system are given. The readings of any visible point in the eye coordinate system are converted to the normalized device coordinates (NDC) in Figure 16(b). In the NDC system, coordinate values range between -1 and 1. Bunny courtesy of the Stanford scanning repository [44].

Table 3: Symbols of 3-D reconstruction and OpenGL applications. In this table, fields in the same column play similar or equivalent roles in both applications.

3-D reconstruction	$\mathbf{I}_{4\times4}$	$\begin{bmatrix}\mathbf{X}\\1\end{bmatrix}$	$\begin{bmatrix}\mathsf{R}_i & \mathbf{t}_i\\0 & 1\end{bmatrix}$	$\begin{bmatrix}\tilde{\mathsf{X}}\\1\end{bmatrix}$	none	$\begin{bmatrix}L_x & 0 & \zeta_0\\0 & L_y & \eta_0\\0 & 0 & 1\end{bmatrix}$
OpenGL	M_{model}	$\begin{bmatrix}x'_o\\y'_o\\z'_o\\w'_o\end{bmatrix}$	M_{view}	$\begin{bmatrix}x_e\\y_e\\z_e\\w_e\end{bmatrix}$	$\begin{bmatrix}x_n\\y_n\\z_n\end{bmatrix}$	none

access $M_{modelview}$ and M_{proj}[2] and then obtain R_i, $\mathbf{t}_i$, L_x, L_y, ζ_0, and η_0 by applying arithmetical operations on the elements of the two matrices.

4.3.1 R_i and $\mathbf{t}_i$

Let's denote the coordinate transformation between $[x_o, y_o, z_o, w_o]^\mathsf{T}$ and $[\mathbf{X}, 1]^\mathsf{T}\begin{bmatrix}\mathsf{R}_w & \mathbf{t}_w\\\mathbf{0}^\mathsf{T} & 1\end{bmatrix}$; in other words, $[x_o, y_o, z_o, w_o]^\mathsf{T} = \begin{bmatrix}\mathsf{R}_w & \mathbf{t}_w\\\mathbf{0}^\mathsf{T} & 1\end{bmatrix}[\mathbf{X}, 1]^\mathsf{T}$. Therefore, by referring to Figures 14 and 15 and comparing sections 4.2 and 2.2, we conclude that $\left[\tilde{\mathsf{X}}, 1\right]^\mathsf{T}$ can be obtained from the following equation:

$$\begin{bmatrix}\tilde{\mathsf{X}}\\1\end{bmatrix} = \begin{bmatrix}\mathsf{R}_e & \mathbf{t}_e\\\mathbf{0}^\mathsf{T} & 1\end{bmatrix} M_{modelview} \begin{bmatrix}\mathsf{R}_w & \mathbf{t}_w\\\mathbf{0}^\mathsf{T} & 1\end{bmatrix}\begin{bmatrix}\mathbf{X}\\1\end{bmatrix},$$

in which

$$\begin{bmatrix}\mathsf{R}_e & \mathbf{t}_e\\\mathbf{0}^\mathsf{T} & 1\end{bmatrix} = \begin{bmatrix}\begin{bmatrix}1 & 0 & 0\\0 & cos(\pi) & -sin(\pi)\\0 & sin(\pi) & cos(\pi)\end{bmatrix} & \begin{bmatrix}0\\0\\0\end{bmatrix}\\\mathbf{0}^\mathsf{T} & 1\end{bmatrix}.$$

[2]By using glGetFloatv(GL_MODELVIEW_MATRIX, modelview_vector) and glGetFloatv(GL_PROJECTION_MATRIX, projection_vector), we can fetch $M_{modelview}$ and M_{proj}, respectively, from the configuration of OpenGL virtual reality and individually store them in arrays modelview_vector and projection_vector.

Readers can understand the reason why $\begin{bmatrix} \mathsf{R}_e & \mathbf{t}_e \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$ is such determined by observing the coordinate relationship between $\left[\tilde{\mathsf{X}}, 1\right]^\mathsf{T}$ and $[x_e, y_e, z_e, w_e]^\mathsf{T}$ in Figures 14 and 15. Therefore,

$$\begin{bmatrix} \tilde{\mathsf{X}} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\pi) & -sin(\pi) \\ 0 & sin(\pi) & cos(\pi) \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ \underbrace{m_{41}}_{0} & \underbrace{m_{42}}_{0} & \underbrace{m_{43}}_{0} & \underbrace{m_{44}}_{1} \end{bmatrix}}_{M_{\text{modelview}}} \overbrace{\begin{bmatrix} \mathsf{R}_w & \mathbf{t}_w \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}}^{[x_o, y_o, z_o, w_o]^\mathsf{T}}$$

$$= \begin{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ -m_{21} & -m_{22} & -m_{23} \\ -m_{31} & -m_{32} & -m_{33} \end{bmatrix} \mathsf{R}_w & \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ -m_{21} & -m_{22} & -m_{23} \\ -m_{31} & -m_{32} & -m_{33} \end{bmatrix} \mathbf{t}_w + \begin{bmatrix} m_{14} \\ -m_{24} \\ -m_{34} \end{bmatrix} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}.$$

Finally, we use the elements of $M_{modelview}$ to represent R_i and $\mathbf{t}_i$, yielding

$$\mathsf{R}_i = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ -m_{21} & -m_{22} & -m_{23} \\ -m_{31} & -m_{32} & -m_{33} \end{bmatrix} \mathsf{R}_w, \tag{36}$$

and

$$\mathbf{t}_i = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ -m_{21} & -m_{22} & -m_{23} \\ -m_{31} & -m_{32} & -m_{33} \end{bmatrix} \mathbf{t}_w + \begin{bmatrix} m_{14} \\ -m_{24} \\ -m_{34} \end{bmatrix}. \tag{37}$$

4.3.2 L_x, L_y, ζ_0, and η_0

Scrutinizing the last two columns in Table 3, we find out that the intrinsic parameters used in 3-D reconstruction applications do not have their counterparts in computer graphics. Likewise, the NDC in computer graphics have no counterparts in 3-D

reconstruction applications, either. In fact, the intrinsic parameters and NDC are related by one type of parameter not shown in Table 3—the width and height of the viewport (denoted by w and h for the following context, respectively).

Points in NDC are eventually rendered on a 2-D viewport for users to visualize. The dimensions of the viewport, which are the width and the height of the API window, can be arbitrarily specified by given API arguments or manual adjustments without any knowledge of NDC. Furthermore,

- $x_n = 0$ is mapped to the horizontal center of the viewport ($\frac{w}{2}$), and $y_n = 0$ is mapped to the vertical center of the viewport ($\frac{h}{2}$);
- x_n serves as the ratio of the horizontal displacement off the viewport center to the half width, and y_n serves as the ratio of the vertical displacement off the viewport center to the half height.

According to the statements, if a point whose NDC is $[x_n, y_n, z_n]^\mathsf{T}$ is rendered on a 2-D viewport at indices $\breve{\boldsymbol{x}}_i = [\breve{x}_i, \breve{y}_i]^\mathsf{T}$, then the relationship between $[x_n, y_n, z_n]^\mathsf{T}$ and $\breve{\boldsymbol{x}}_i = [\breve{x}_i, \breve{y}_i]^\mathsf{T}$ is as follows:

$$\breve{\boldsymbol{x}}_i = \begin{bmatrix} \breve{x}_i \\ \breve{y}_i \end{bmatrix} = \begin{bmatrix} \frac{w}{2} & 0 & x_o + \frac{w}{2} \\ 0 & \frac{h}{2} & y_o + \frac{h}{2} \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}. \tag{38}$$

In equation (38), x_o and y_o are the origin coordinates of the viewport, so they are usually 0. With the fact that NDC values are independent of w and h, this equation implies that users can change the ratio of x to y of a rendered object in the viewport by manually changing the ratio of w to h.

Note that $[\breve{x}_i, \breve{y}_i]^\mathsf{T} = [\hat{x}_i, h - \hat{y}_i]^\mathsf{T}$, as illustrated in Figures 14 and 15, thus

$$\breve{y}_i = \frac{h}{2} y_n + (y_o + \frac{h}{2}) = h - \hat{y}_i = h - L_y \frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i}.$$

Based on equations (33) and (34), we obtain $y_n = -\frac{n}{tz_e}y_e$, so

$$L_y\frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} = h - [\frac{h}{2}y_n + (y_o + \frac{h}{2})] = h - [-\frac{hn}{2tz_e}y_e + (y_o + \frac{h}{2})] = \frac{hn}{2t}\frac{y_e}{z_e} + (\frac{h}{2} - y_o).$$

Therefore,

$$\begin{cases} L_y = & \frac{hn}{2t} \\ \eta_o = & \frac{h}{2} - y_o \end{cases}.$$

Likewise,

$$\begin{cases} L_x = & \frac{wn}{2r} \\ \zeta_o = & x_o + \frac{w}{2} \end{cases}.$$

After we plug the results in equation (35) to substitute for n, t, and r in the equations above, we represent L_x, L_y, ζ_0, and η_0 in terms of the elements of M_{proj}, so we obtain

$$\begin{cases} L_x = & \frac{w\left((M_{proj})_{11}\right)}{2\left((M_{proj})_{13}+1\right)} \\ L_y = & \frac{h\left((M_{proj})_{22}\right)}{2\left((M_{proj})_{23}+1\right)} \\ \zeta_0 = & \frac{w}{2} \\ \eta_0 = & \frac{h}{2} \end{cases}. \tag{39}$$

4.4 Synchronous videos

Generating synchronous videos to simulate the outputs of multiple synchronized cameras requires advanced OpenGL knowledge. An OpenGL system has three types of buffers [43]: stencil buffer, depth buffer, and color buffers.[3] Among all these buffers, the color buffers contain the renderings—the drawings in the windows shown on the screen, such as Figure 16(c), and the renderings can be fetched through an OpenGL API—glReadPixels. To guarantee that the shape of the drawing approximate real ocean surface, we generate the height and texture using the Fast-Fourier-Transform-based (FFT) Ocean Simulation (supported by NVIDIA [39], based on the work of

[3]In practice, the number of color buffers is determined by users. It can be one or more.

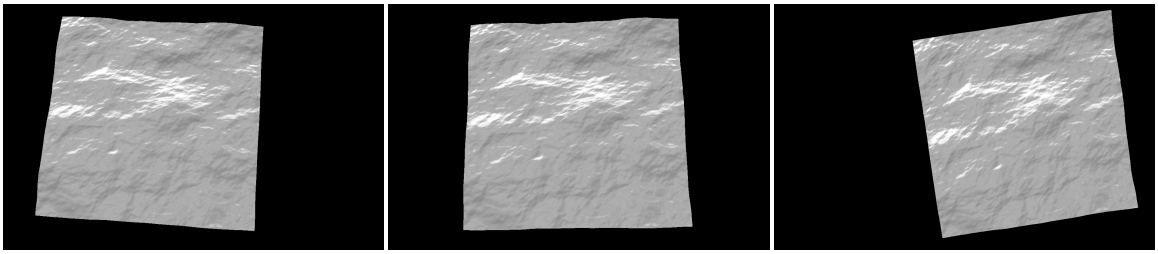


Figure 17: A synthetic ocean surface observed from different positions. Note that the black regions have no effects on the reconstruction and calibration refinement since the region of interest for reconstruction is selected to be an internal portion of the observed patch.

Tessendorf [47]).

The observation positions, depending on $M_{modelview}$s and M_{proj}s, are manually set at the very beginning of generating the video sequences. Stored $M_{modelview}$s and M_{proj}s are repetitively loaded to replace OpenGL parameters, GL_MODELVIEW and GL_PROJECTION, respectively, so that the OpenGL pipeline renders the observations from those positions into the color buffers.[4] Figure 17 illustrates one example of a deforming synthetic ocean surface observed from different positions at a specific moment.

4.5 Algorithm validation using synthetic data

In the previous chapter, the effectiveness of the proposed calibration refinement algorithm is validated using real data. However, the space-time model of a real ocean surface and the true camera parameters of a stereo computer vision rig are difficult to be obtained, so we conduct experiments on synthetic data in this section. Although the synthetic data generated through OpenGL cannot fully approximate all physical properties of real datasets, by using the coordinate transformation formulas derived in previous sections of this chapter, we have the flexibility to quantify and control

[4]Note that some commercial software and graphics cards may be equipped with the functions of stereo rendering, in which stereo observations can be simultaneously fetched from different color buffers. However, this type of technique currently supports dual renderings only.

various types of camera parameters of synthetic data. Then, we can simulate the situations in which the camera parameters are corrupted and validate the correctness of the derived gradient vectors in Tables 1 and 2.

Five experiments are conducted in this section. The target object for reconstruction is a piece of ocean surface produced through inverse CUDA FFT, and the observation positions are configured as shown in Figure 17. Corresponding true camera parameters, intrinsic and extrinsic, are generated through equations (36), (37), and (39). Then, artificial errors are introduced to these true parameters to produce corrupted ones. These corrupted camera parameters along with the visual observations are used as the inputs of our algorithms, generating elevation maps for the comparisons to the ground truth.

The introduced relative errors used to corrupt the true translation vectors are listed in the second row of Table 4. Since translation vectors are three-by-one vectors and because three observation positions are configured, nine numbers are listed in the row. The joint operations of the calibration refinement and the 3-D reconstruction algorithms produce an elevation map shown in Figure 19(a). For comparison purposes, we also produce an elevation map using variational 3-D reconstruction alone and display it in Figure 19(c). The differences between the ground truth and the resulting elevation maps are shown in Figures 19(b) and 19(d). Finally, because the gradient descent method is utilized to decrease E_{calib} as a means of parameter refinement, we show how E_{calib} changes during the iterations of calibration refinement in the figure at the bottom. Likewise, the numbers in the third row of Table 4, the second row of Table 5, and the third row of Table 5 are appointed to corrupt true parameters of rotation axises, focal lengths, and principal point positions, respectively. Corresponding experiment results and their explanations are illustrated in Figures 20, 21, and 22.

Table 4: Relative errors for corrupting the true extrinsic camera parameters

	cam1			cam2			cam3		
$\mathbf{t}$	-2.38%	2.51%	0.005%	1.05%	-3.38%	1.6%	1.35%	-1.17%	0.6%
$\vec{\omega}$	-1.86%	-2.32%	-2.46%	1.98%	-1.85%	-0.82%	-0.32%	2.06%	-2.02%

Table 5: Relative errors for corrupting the true intrinsic camera parameters

	cam 1		cam 2		cam 3	
$\left(L_x, L_y\right)^\mathsf{T}$	1.23%	-2.44%	-0.32%	-0.60%	1.03%	-1.22%
$\left(\zeta_0, \eta_0\right)^\mathsf{T}$	1.76%	-1.01%	2.45%	-2.34%	-2.43%	0.39%

In these experiments, the magnitudes of the introduced relative errors are approximately 2%, and their influence on the reconstruction is conspicuous. From these experiments, we observe that the variational 3-D reconstruction algorithm alone is incapable of generating a reliable 3-D model given miscalibrated cameras whose parameters deviate from the true values by 2%: most of reconstructions under this situation do not resemble the ground truth at all. By contrast, the performance of the joint operations of the calibration refinement and 3-D reconstruction algorithms is clearly promising. The generated elevation maps approximate the ground truth, so their error maps (point-wise height difference between the ground truth and the elevation maps) exhibit values around zero.

Recall that we normalized 3-D space and image domains using two similarity matrices to produce a composite gradient descent vector for the simultaneous reduction of E_{calib} with respect to multiple types of camera parameters in section 3.5. Here, an experiment is particularly conducted to validate the derivations in section 3.5. In this experiment, we only focus on the extrinsic camera parameters and corrupt them using the relative errors in Table 6.[5] Given that this experiment uses more corrupted parameters than others do, the magnitudes of the errors in Table 6 are set to be smaller than those in Tables 4 and 5.

Note that the ultimate values of the refined camera parameters are not particularly

[5]The reason why we only test the extrinsic parameters is explained in the following chapter.

Table 6: Relative errors for corrupting the true extrinsic camera parameters. Note that this experiment is conducted for the validation of section 3.5.

	cam 1			cam 2			cam 3			cam 4		
t	-1.04%	-0.63%	-1.13%	-0.39%	1.42%	0.29%	1.19%	-0.09%	0.84%	-0.27%	-1.33%	0.66%
$\vec{\omega}$	0.02%	-0.38%	-0.86%	-0.74%	-0.28%	0.8%	0.23%	-0.64%	-1.01%	-1.43%	0.75%	0.46%

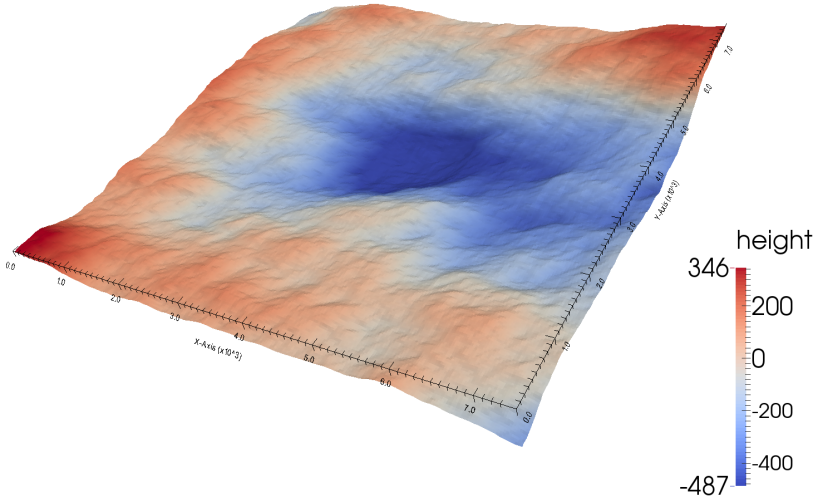
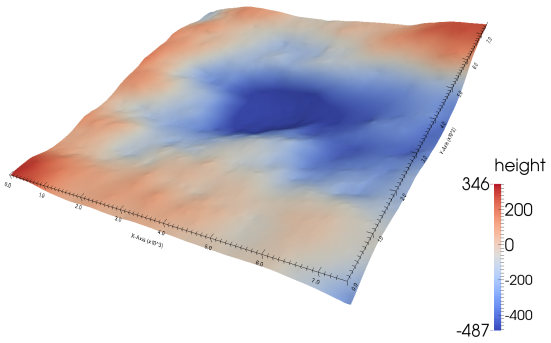


Figure 18: Elevation map (ground truth)

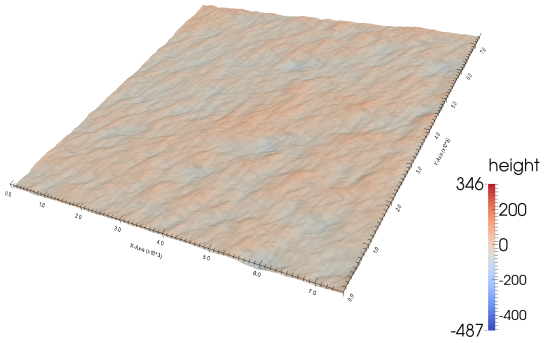
listed for the comparisons to the true ones for the following two reasons:

- In most 3-D reconstruction applications, the definition of camera parameter refinement is usually given based on the improvement of the reconstruction.
- The refinement of the camera parameters are dependent on how a local minimum along a chosen descent direction is obtained. Different numerical line search methods, different local minimum finders, or even the ratio of the iterations of the calibration refinement to the iterations of reconstruction affects the ultimate values of the refined camera parameters.

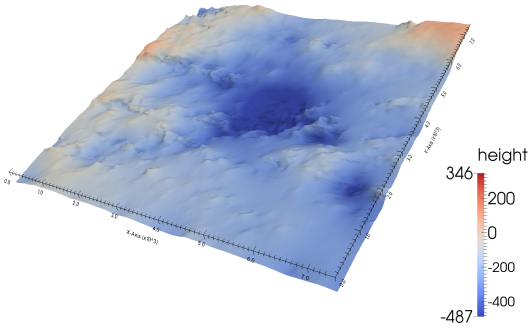
As a result, we did not spend efforts tracking how the camera parameters are eventually changed.



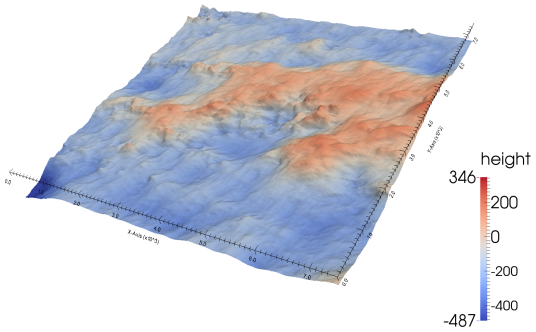
(a) The elevation map produced through the joint operations of the calibration refinement and variational 3-D reconstruction algorithms



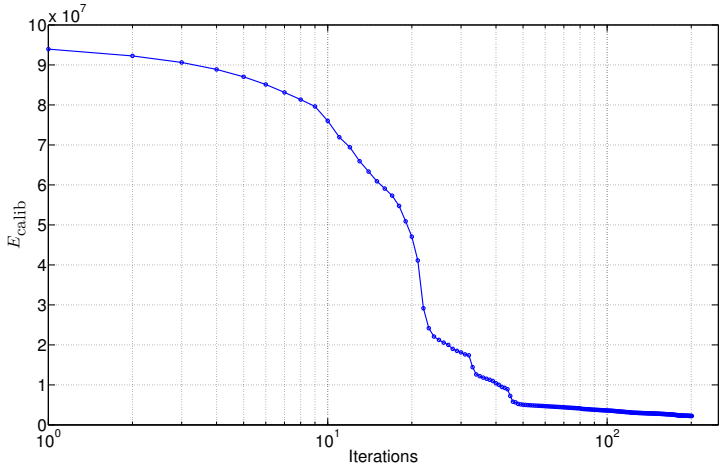
(b) The difference between Figures 18 and 19(a)



(c) The elevation map produced through the variational 3-D reconstruction algorithm alone

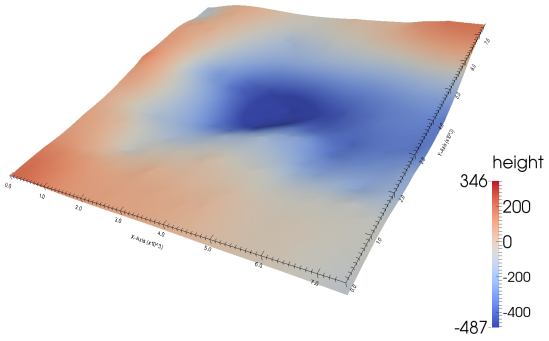


(d) The difference between Figures 18 and 19(c)

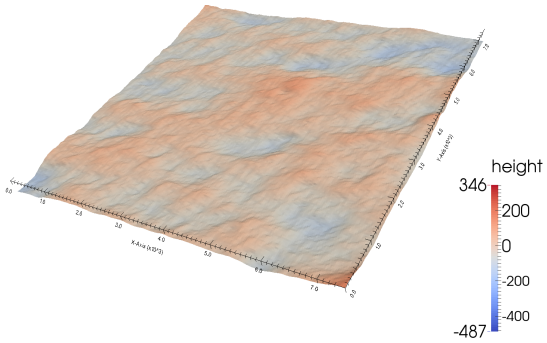


(e) The change of E_{calib} (equation (17)) during the process of calibration refinement

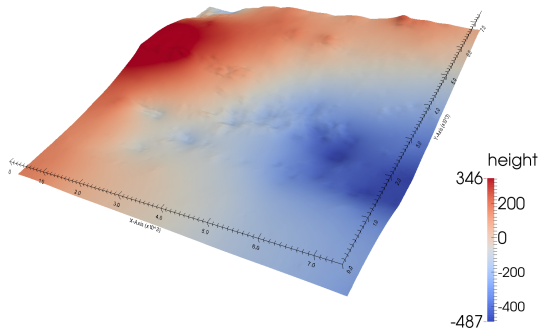
Figure 19: Validation of the refinement of **t**



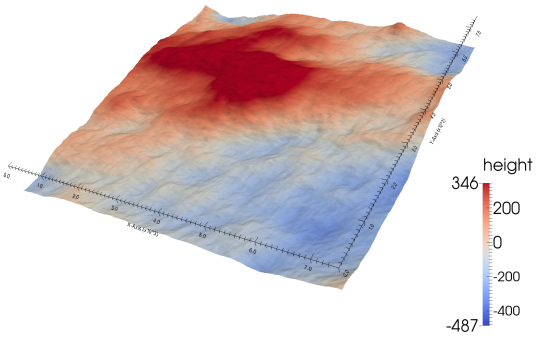
(a) The elevation map produced through the joint performance of the calibration refinement and variational 3-D reconstruction algorithms



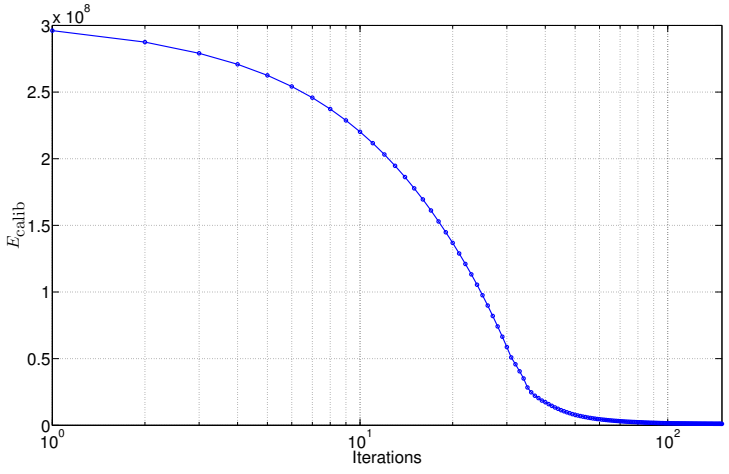
(b) The difference between Figures 18 and 20(a)



(c) The elevation map produced through the variational 3-D reconstruction algorithm alone

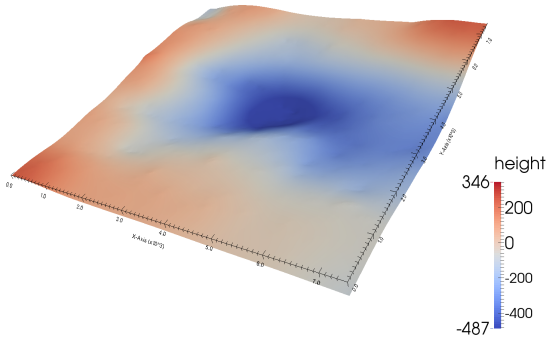


(d) The difference between Figures 18 and 20(c)

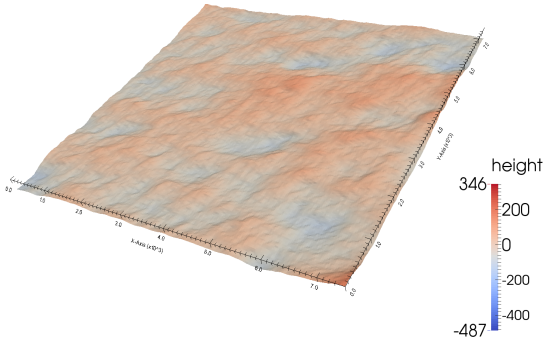


(e) The change of E_{calib} during the process of calibration refinement

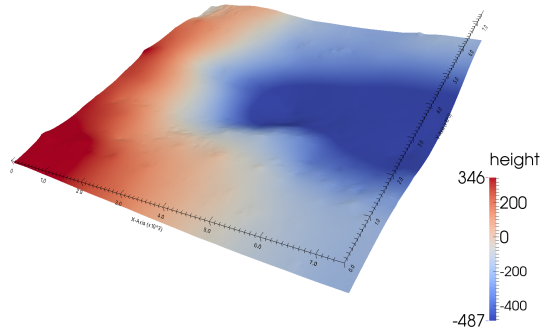
Figure 20: Validation of the refinement of $\vec{\omega}$



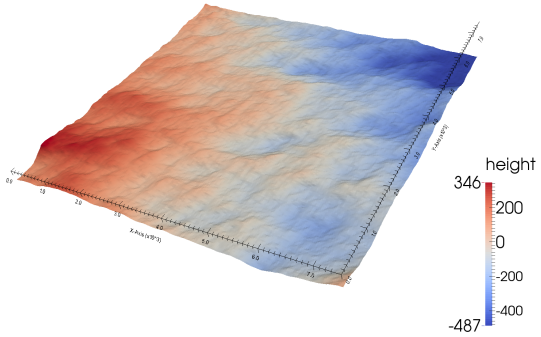
(a) The elevation map produced through the joint operations of the calibration refinement and variational 3-D reconstruction algorithms



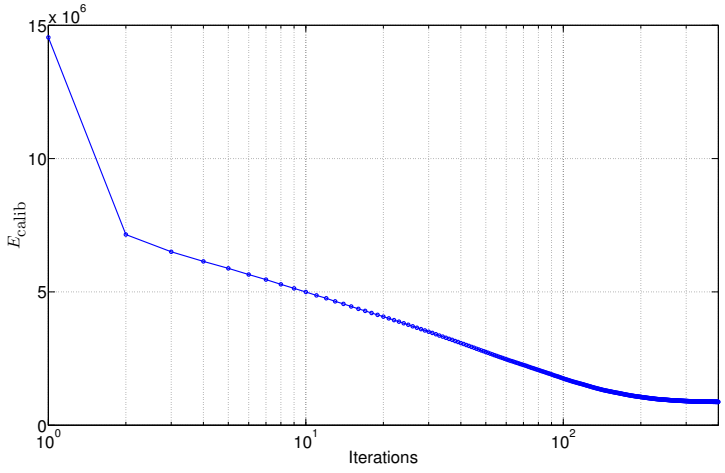
(b) The difference between Figures 18 and 21(a)



(c) The elevation map produced through the variational 3-D reconstruction algorithm alone

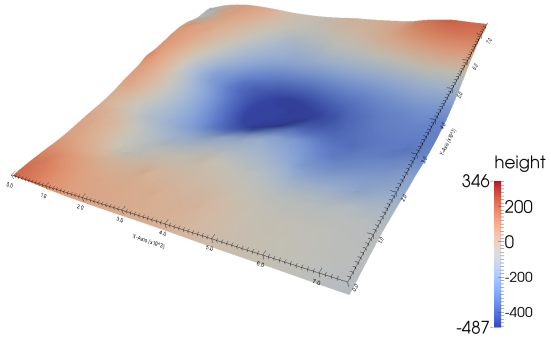


(d) The difference between Figures 18 and 21(c)

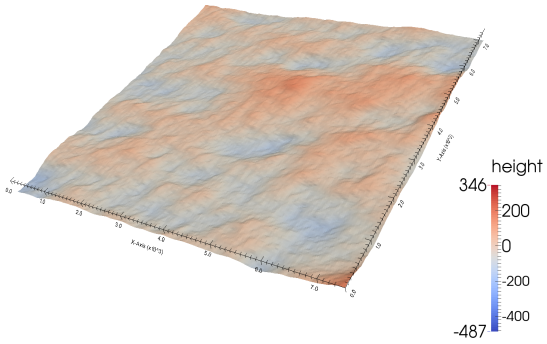


(e) The change of E_{calib} during the process of calibration refinement

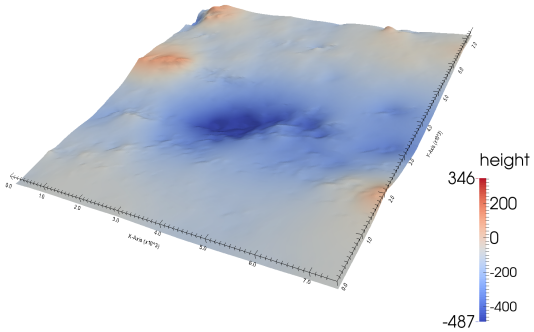
Figure 21: Validation of the refinement of $\left(L_x, L_y\right)^{\mathsf{T}}$



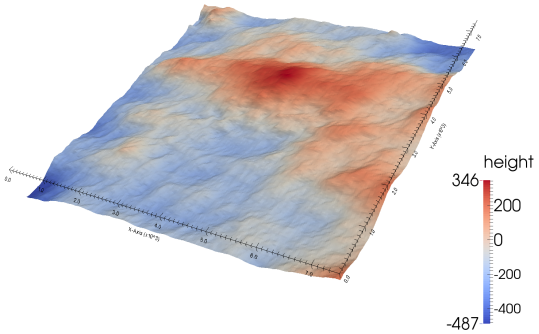
(a) The elevation map produced through the joint operations of the calibration refinement and variational 3-D reconstruction algorithms



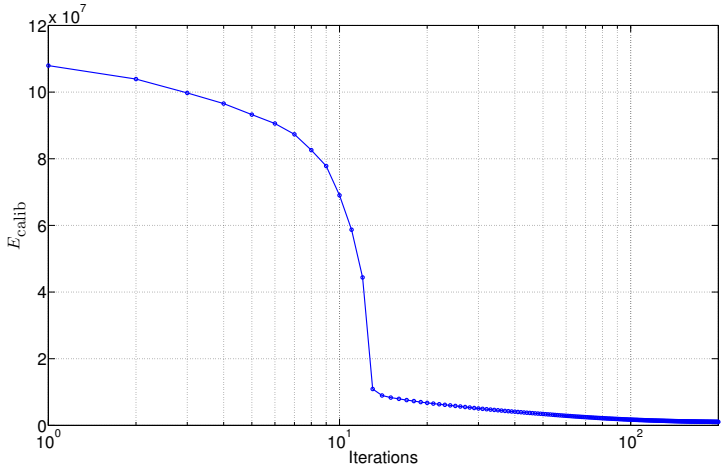
(b) The difference between Figures 18 and 22(b)



(c) The elevation map produced through the variational 3-D reconstruction algorithm alone

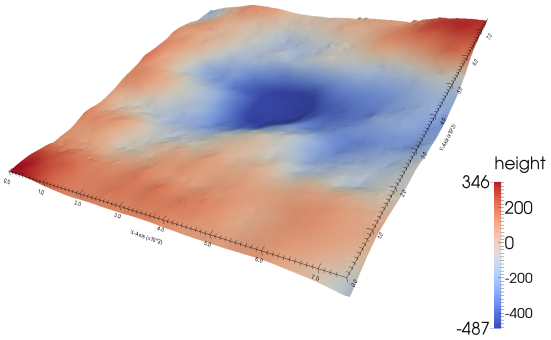


(d) The difference between Figures 18 and 22(c)

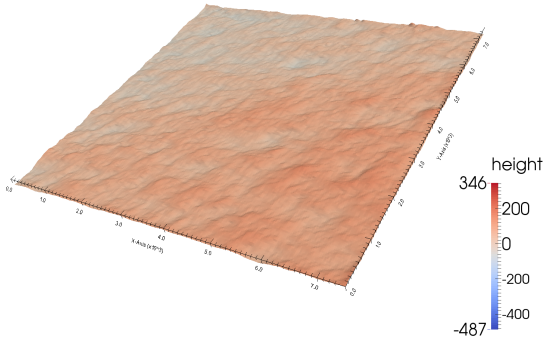


(e) The change of E_{calib} during the process of calibration refinement

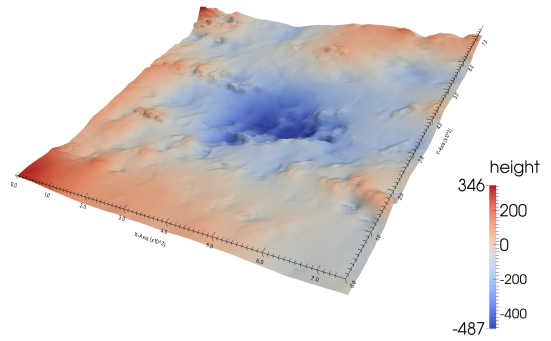
Figure 22: Validation of the refinement of $\left(\zeta_0, \eta_0\right)^{\mathsf{T}}$



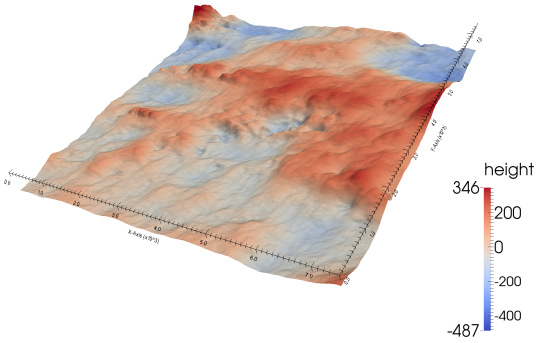
(a) The elevation map produced through the joint operations of the calibration refinement and variational 3-D reconstruction algorithms



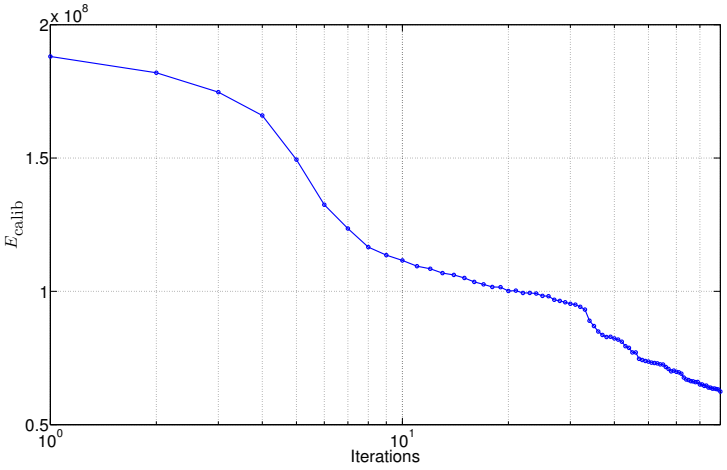
(b) The difference between Figures 18 and 23(a)



(c) The elevation map produced through the variational 3-D reconstruction algorithm alone



(d) The difference between Figures 18 and 23(c)



(e) The change of E_{calib} during the process of calibration refinement

Figure 23: Validation of the joint refinement of $\mathbf{t}$ and $\vec{\omega}$

CHAPTER V

SPACE-TIME CAMERA CALIBRATION REFINEMENT

In applications that 4-D (space-time) analyses are required to validate oceanographic theories, stacking a series of variational 3-D reconstructions independently generated for each time moment to emulate a 4-D model seems a straightforward method. However, this type of "makeshift" 4-D model is less numerically robust and more computationally expensive than the "4-D reconstruction," which is built taking into consideration both spatial and temporal coherence [22].

Still, the accuracy of the 4-D reconstruction from [22] is subject to the estimations of camera parameters. In most outdoor applications, the camera parameters may be unpredictably perturbed by natural factors such as wind and vibrations. Consequently, because of the temporal coherence imposed during the 4-D reconstruction, errors of the camera parameters at a certain moment will not only undermine the reconstruction accuracy related to the moment but also deteriorate its temporal neighbor. Therefore, removing the unpredictable perturbations on the camera parameters, under the condition that the inception and the duration of the external influence are unknown, is necessary for an accurate 4-D reconstruction. Seemingly, we may sequentially apply the calibration refinement method developed in the previous sections on each 3-D model extracted from each moment of the 4-D reconstruction. However, in this case, each 3-D reconstruction of a moment is jointly performed with the calibration refinement method without the concerns of maintaining the temporal coherence between 3-D reconstructions. Consequently, this strategy (although straightforward) is prone to break the temporal coherence deliberately added to perform the space-time reconstruction of a ocean surface.

The best strategy is to formulate the refinement of the camera parameters and the 4-D reconstruction of the ocean surface into an energy functional that measures 1) the error between captured videos and the reprojection of the 4-D reconstruction onto the snapshots of the videos, 2) spatial and temporal smoothness of the reconstruction, and 3) the variance of the perturbed camera parameters. The first two conditions are satisfied in the energy functional proposed in [22]. In this chapter, we explore what types of regularizers or priors are appropriate for the joint 4-D reconstruction and calibration refinement in a period of time.

5.1 Variational 4-D reconstruction of sea states

The energy functional for the variational 4-D reconstruction was established by Gallego [21, 22] through the modifications of equations (6): Temporal coherence is imposed on all terms in equation (5) , so E_{data}, E_{geom}, and E_{rad} become

$$\begin{cases} E_{data} = \int_{\mathbb{T}} \sum_{i=1}^{N} \left\{ \int_{R_i} \frac{1}{2} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}} \right\} d\tau \\ E_{geom} = \int_{\mathbb{T}} \int_{U} \frac{1}{2} (Z_u^2 + Z_v^2 + Z_\tau^2) d\boldsymbol{u} d\tau \\ E_{rad} = \int_{\mathbb{T}} \int_{U} \frac{1}{2} (f_u^2 + f_v^2 + f_\tau^2) d\boldsymbol{u} d\tau \end{cases} . \tag{40}$$

Compared to the corresponding definitions in the variational 3-D reconstruction (equation (6)), Z and f are now functions of three variables (u, v, τ) with domain in $U_T \coloneqq U \times [0, \mathbb{T}]$, where $\mathbf{u} = (u, v)^\top \in U$ are spatial variables and $\tau \in [0, \mathbb{T}]$ is the temporal variable. E_{data} now quantifies the discrepancies between the 4-D model and the captured videos. The derivatives of Z and f with respect to time are added to E_{geom} and E_{rad}, respectively, to represent their temporal smoothness. From an algebraic perspective, the minimizers of this energy functional are functions of not only U but also $\mathbb{T}$; from a geometric perspective, the minimizers change to a manifold of the graphs.

5.2 Calibration refinement for variational 4-D reconstruction

Recall that the calibration refinement algorithm developed so far is concerned with 3-D variational reconstruction. Now that we would like to generalize the calibration refinement algorithm to be applicable to the 4-D reconstruction, we need to revise the notation introduced in chapter 2. In the following context, superscript τ is added to the extrinsic parameters and relative terms to underline that the terms are estimated at moment τ. The intrinsic parameters are exempt from the notation modification because they are assumed to remain constant while the extrinsic parameters are being corrupted. In addition, many symbols, such as $\hat{\mathbf{x}}$, $\mathbf{x}$, π_i, and $\tilde{\mathbf{X}}_i$, are also exempt from superscript τ because they are not used to represent the final results of the derivations.

Since environmental factors such as breeze or vibrations smoothly influence the extrinsic camera parameters, we can assume that the temporal changes of the camera parameters are also smooth, which can be implemented by adding one term imposing temporal smoothness constraints on the camera parameters into equation (5). Hence, the energy functional for joint 4-D reconstruction and calibration refinement is designed as follows:

$$E(f, Z, \lambda) = E_{data}(f, Z, \lambda) + \alpha E_{geom}(Z) + \beta E_{rad}(f) + \gamma E_{cam}(\lambda), \tag{41}$$

where $\alpha, \beta, \gamma > 0$. In this equation, E_{geom} and E_{rad} are identical to the ones in equation (40). E_{data} is redefined as

$$\begin{aligned} E_{data} = \quad & \int_{\mathbb{T}} \sum_{i=1}^{N_c} \left\{ \int_{R_i} \tfrac{1}{2} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}} \right\} d\tau \\ + \quad & \tfrac{W}{2} \int_{\mathbb{T}} \left\{ \sum_{i=1}^{N_c - 1} \sum_{j=i+1}^{N_c} (d_{ij}^{\tau} - d_{ij}^{o})^2 \right\} d\tau, \end{aligned}$$

and E_{cam}[1] is defined as

$$E_{cam} = \frac{1}{2} \int_{\mathbb{T}} \psi(\lambda, \lambda_\tau) d\tau. \tag{42}$$

[1]This representation is a general statement: ψ may depend on λ, λ_τ, or both.

The minimizers of equation (41) are determined by the zero functional derivatives with respect to the arguments (denoted by $\frac{\delta E}{\delta Z} = 0$, $\frac{\delta E}{\delta f} = 0$, and $\frac{\delta E}{\delta \lambda} = 0$). In other words,

$$D_{(h,g,\eta)}E(Z,f,\lambda) \triangleq \frac{\delta E}{\delta(Z,f,\lambda)}(h,g,\eta) = \frac{d}{d\epsilon}E\Big((Z,f,\lambda) + \epsilon(h,g,\eta)\Big)\Big|_{\epsilon=0} = 0.$$

The methods of deriving the analytical forms of $\frac{\delta E}{\delta Z} = 0$ and $\frac{\delta E}{\delta f} = 0$ are explored in [21, p.100]. Note that although the energy functional in [21, p.100] differs from that in (41), $\frac{\delta E}{\delta Z} = 0$ and $\frac{\delta E}{\delta f} = 0$ form the same set of PDEs because the second term of E_{data} and E_{cam} do not depend on Z or f.

To compute $\frac{\delta E}{\delta \lambda}$, we augment λ with an artificial time variable t and differentiate λ with respect to t using the chain rule, yielding $\frac{\partial E}{\partial t} = \left\langle \frac{\delta E}{\delta \lambda}, \lambda_t \right\rangle_{L^2(\mathbb{T})}$, where $\langle \cdot, \cdot \rangle_{L^2(\mathbb{T})}$ is the L^2-inner product operator between functions defined over the interval of duration $\mathbb{T}$. $\frac{\partial E}{\partial t} = \left\langle \frac{\delta E}{\delta \lambda}, \lambda_t \right\rangle_{L^2(\mathbb{T})}$ lays down the path toward obtaining an extremum of E by evolving λ with respect to t using a descent method.

Thus, differentiating E with respect to t leads to

$$\frac{\partial E}{\partial t} = \frac{\partial E_{data}}{\partial t} + \frac{\partial E_{cam}}{\partial t}, \tag{43}$$

where the smoothness terms in E_{geom} and E_{rad} vanish because they do not depend on the camera parameters λ.

5.2.1 $\frac{\partial E_{data}}{\partial t}$

Since R_i^τ and $\mathbf{t}_i^\tau$ are the parameters of camera i, we can drop the Σ symbol when differentiating E_{data} with respect to t. Thus, $\frac{\partial E_{data}}{\partial t} =$

$$\int_{\mathbb{T}} \frac{\partial}{\partial t}\Big\{ \int_{R_i} \frac{1}{2}\Big[I_i\big(\hat{\mathbf{x}}\big) - f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big)\Big]^2 d\hat{\mathbf{x}}\Big\} d\tau \tag{44}$$

$$+ \int_{\mathbb{T}} \frac{\partial}{\partial t}\Big\{ \frac{W}{2} \sum_{j=1(j\neq i)}^{N_c} (d_{ij}^\tau - d_{ij}^o)^2 \Big\} d\tau. \tag{45}$$

The further derivations of the two terms are similar to the contents in appendix A and section 3.3, so we omit the tedious deduction and directly show the results here.

Term (44) can be written as

$$\Big\langle \lambda_t(\tau), \int_{\partial U} \frac{\Big[I_i\big(\pi_i(\mathbf{X}^\tau)\big) - f(\mathbf{X}^\tau)\Big]^2}{2} \big\langle \frac{\partial\big(\pi_i(\mathbf{X}^\tau)\big)}{\partial\lambda}, Q \frac{\partial\big(\pi_i(\mathbf{X}^\tau)\big)}{\partial s} \big\rangle ds + \int_U \Big[f(\mathbf{X}^\tau) - I_i\big(\pi_i(\mathbf{X}^\tau)\big)\Big] \big\langle \frac{\partial \mathbf{X}^\tau}{\partial\lambda}, \nabla_{\mathbf{s}} f\big(\mathbf{X}^\tau\big)\big\rangle J_i d\boldsymbol{u} \Big\rangle_{L^2(\mathbb{T})}, \tag{46}$$

in which superscript τ is deliberately added to $\mathbf{X}^\tau$ to emphasize that moment of the observation. Term (45) is eventually represented as

$$\Big\langle \lambda_t(\tau), W \sum_{j=1(j\neq i)}^{N_c} \Big(1 - \frac{d_{ij}^o}{\|\Omega\|}\Big) \big\langle \frac{\partial\Omega}{\partial\lambda}, \Omega \big\rangle \Big\rangle_{L^2(\mathbb{T})}. \tag{47}$$

In this equation, $\Omega = \big(\mathbf{t}_i^\tau - \mathsf{R}_i^\tau (\mathsf{R}_j^\tau)^\mathsf{T} \mathbf{t}_j^\tau\big)$.

5.2.2 $\frac{\partial E_{cam}}{\partial t}$ if ψ is defined as $\psi = \|\lambda_\tau\|^2$

The most straightforward choice for ψ that is capable of representing the smoothness of the temporal changes of λ is the derivative, so ψ in (42) is chosen to be $\|\lambda_\tau\|^2$. With $\psi = \|\lambda_\tau\|^2$, $\frac{\partial E_{cam}}{\partial t}$ becomes

$$\frac{\partial}{\partial t}\frac{1}{2}\int_{\mathbb{T}} \|\lambda_\tau\|^2 d\tau = \int_{\mathbb{T}} \|\lambda_\tau\| \underbrace{\|\lambda_\tau\|_t}_{\frac{\langle \lambda_{\tau t}, \lambda_\tau\rangle}{\|\lambda_\tau\|}} d\tau = \int_{\mathbb{T}} \langle \lambda_{t\tau}, \lambda_\tau\rangle \, d\tau = \langle \lambda_t, \lambda_\tau\rangle \Big|_0^{\mathbb{T}} - \int_{\mathbb{T}} \langle \lambda_t, \lambda_{\tau\tau}\rangle \, d\tau.$$

We can assume that λ is not corrupted at 0 and $\mathbb{T}$; thus, we can set zero boundary conditions, equivalently $\lambda_t\Big|_{\mathbb{T}} = \lambda_t\Big|_0 = 0$. In this case,

$$\frac{\partial E_{cam}}{\partial t} = \Big\langle \lambda_t(\tau), -\gamma \lambda_{\tau\tau}^\mathsf{T} \Big\rangle_{L^2(\mathbb{T})}. \tag{48}$$

Therefore, by setting

$$\lambda_t = \gamma \lambda_{\tau\tau}^\mathsf{T}, \tag{49}$$

we obtain the gradient descent vector to decrease E_{cam}. Note that equation (49) is of the form of the heat equation, and numerically solving this partial differential equation requires the Von Neumann stability analysis [26] to restrict the ratio between the step sizes of t and τ.

The advantage of this constraint is its simple derivation. However, a significant drawback of this constraint is that the evaluation of $\psi\Big|_{\tau_0}$ only numerically depends on (τ_0-1), τ_0, and (τ_0+1). When the data-acquiring rate (the frame rate of the camera) is high compared to the fundamental frequency of the temporal changes of camera parameters, the constraint imposed by E_{cam} on λ brings relatively local smoothness effect to the temporal distribution of λ. As a result, this method is not practical in many applications since users can not accordingly adjust the influence region of the constraint based on the temporal properties of the λ.

5.2.3 $\frac{\partial E_{cam}}{\partial t}$ if ψ is defined as $\psi = \frac{\|\lambda-\mu\|^2}{\mathbb{T}}$

In the cases that the camera parameters are occasionally and briefly perturbed and are otherwise intact during a time interval, we can set $\psi = \frac{(\lambda-\mu)^2}{\mathbb{T}}$ so that $E_{cam} = \frac{1}{2}\int_{\mathbb{T}} \frac{(\lambda-\mu)^2}{\mathbb{T}} d\tau$ represents the overall variance of the camera parameters within the interval. Therefore, $\frac{\partial E_{cam}}{\partial t}$ becomes

$$
\begin{aligned}
&\frac{\partial}{\partial t}\frac{\gamma}{2}\int_{\mathbb{T}} \frac{\|\lambda-\mu\|^2}{\mathbb{T}} d\tau \\
=&\frac{1}{\mathbb{T}}\frac{\gamma}{2}\int_{\mathbb{T}} 2(\lambda-\mu)^{\mathsf{T}}(\lambda_t-\mu_t) d\tau \\
=&\frac{1}{\mathbb{T}}\gamma\int_{\mathbb{T}} (\lambda-\mu)^{\mathsf{T}}(\lambda_t - \frac{\partial}{\partial t}\int_{\mathbb{T}}\frac{\lambda}{\mathbb{T}} d\tau) d\tau \\
=&\frac{\gamma}{\mathbb{T}}\int_{\mathbb{T}} (\lambda-\mu)^{\mathsf{T}}(\lambda_t - \frac{1}{\mathbb{T}}\int_{\mathbb{T}}\lambda_t d\tau) d\tau \\
=&\frac{\gamma}{\mathbb{T}}\int_{\mathbb{T}} (\lambda-\mu)^{\mathsf{T}}\lambda_t d\tau - \frac{\gamma}{\mathbb{T}}\int_{\mathbb{T}} (\lambda-\mu)^{\mathsf{T}}\Big(\frac{1}{\mathbb{T}}\int_{\mathbb{T}}\lambda_t d\tau\Big) d\tau \\
=&\frac{\gamma}{\mathbb{T}}\int_{\mathbb{T}} \lambda_t^{\mathsf{T}}(\lambda-\mu) d\tau - \cancel{\Big[\frac{\gamma}{\mathbb{T}}\int_{\mathbb{T}} (\lambda-\mu)^{\mathsf{T}} d\tau\Big]}\Big(\frac{1}{\mathbb{T}}\int_{\mathbb{T}}\lambda_t d\tau\Big) \\
=&\left\langle \lambda_t, \frac{\gamma}{\mathbb{T}}(\lambda-\mu) \right\rangle_{L^2(\mathbb{T})}.
\end{aligned}
\tag{50}
$$

Compared to $\psi = \|\lambda_\tau\|^2$, the most obvious improvement of this design is its computation complexity: Since the analytical form of equation (50) carries no differentials, users do not have to employ numerical schemes to evaluate the corresponding gradient descent vector, which indicates the Von Neumann stability test for obtaining

the optimal step sizes of τ and t is not required. However, the disadvantage of this method is that it can only be applied to the situation in which the duration of the perturbation is relatively short compared to the overall length of the time interval.

5.2.4 $\frac{\partial E_{cam}}{\partial t}$ if ψ is defined as $\psi = \int_{\tau-w}^{\tau+w} \frac{\|\lambda - \left({}^{w}\mu^{\tau}\right)\|^2}{2w} dy$, where $\left({}^{\tau}\mu^{w}\right) = \frac{1}{2w}\int_{\tau-w}^{\tau+w} \lambda(y)dy$

The two designs of ψs above have their individual advantages and disadvantages, which prompt us to design a new E_{cam} so that we can have the flexibility to control the range of the smooth constraint. Therefore, we conceive

$$E_{cam} = \frac{\gamma}{2}\int_{\mathbb{T}}\int_{\tau-w}^{\tau+w} \frac{\|\lambda - \left({}^{w}\mu^{\tau}\right)\|^2}{2w} dy d\tau, \tag{51}$$

in which

$$\left({}^{\tau}\mu^{w}\right) = \frac{1}{2w}\int_{\tau-w}^{\tau+w} \lambda(y)dy. \tag{52}$$

In equation (52), $\left({}^{\tau}\mu^{w}\right)$ represents the local mean of λ evaluated within duration $[\tau - w, \tau + w]$, so the integrand of the outer integral of equation (51), $\int_{\tau-w}^{\tau+w} \frac{\|\lambda - \left({}^{w}\mu^{\tau}\right)\|^2}{2w} dy$, represents the local variance of λ in $[\tau - w, \tau + w]$.

The derivation of $\frac{\partial E_{cam}}{\partial t}$ is separately elucidated in the appendix C because it is long and complicated. In conclusion, if the gradient descent method is used to attain a local minimum of E_{cam}, we should set

$$\lambda_t(\tau) \;=\; -\gamma\left\{\lambda(\tau) - \frac{1}{4w^2}\int_{\tau-w}^{\tau+w}\left(\int_{x-w}^{x+w}\lambda dz\right)dx\right\}. \tag{53}$$

Therefore, using the results in equations (46), (47), and (53), we obtain the gradient descent flow for attaining $\frac{\delta E}{\delta \lambda} = 0$ as follows:

$$\begin{aligned}\lambda_t(\tau) = -\Bigg\{ & \int_{\partial U} \frac{\left[I_i\big(\pi_i(\mathbf{X}^{\tau})\big) - f(\mathbf{X}^{\tau})\right]^2}{2} \Big\langle \frac{\partial\big(\pi_i(\mathbf{X}^{\tau})\big)}{\partial\lambda}, Q\frac{\partial\big(\pi_i(\mathbf{X}^{\tau})\big)}{\partial s}\Big\rangle ds \\ & + \int_{U}\Big[f(\mathbf{X}^{\tau}) - I_i\big(\pi_i(\mathbf{X}^{\tau})\big)\Big]\Big\langle\frac{\partial\mathbf{X}^{\tau}}{\partial\lambda}, \nabla_{\mathfrak{s}} f\big(\mathbf{X}^{\tau}\big)\Big\rangle J_i d\boldsymbol{u} \\ & + W\sum_{j=1(j\neq i)}^{N_c}\Big(1 - \frac{d_{ij}^{o}}{\|\Omega\|}\Big)\Big\langle\frac{\partial\Omega}{\partial\lambda}, \Omega\Big\rangle + \gamma\lambda(\tau) - \frac{\gamma}{4w^2}\int_{\tau-w}^{\tau+w}\left(\int_{x-w}^{x+w}\lambda dz\right)dx\Bigg\}.\end{aligned} \tag{54}$$

5.2.5 Interpretation of the local variance prior

The implementation of the the second term of equation (53) demonstrates the smoothness effect it applies to λ. $\frac{1}{4w^2}\int_{\tau_o-w}^{\tau_o+w}\big(\int_{x-w}^{x+w}\lambda dz\big)dx$ is equivalent to the L^2 inner product of λ and the red triangle shown in Figure 24(a) within the interval of $[\tau_o-2w, \tau_o+2w]$. This triangle is an equilateral triangle with an unity height and a base length of $4w$ and is denoted by ${}^{w}\Lambda^{\tau_0}$[2]. As a result, the effect of $\frac{1}{4w^2}\int_{\tau-w}^{\tau+w}\big(\int_{x-w}^{x+w}\lambda dz\big)dx$ on λ during $\mathbb{T}$ is the convolution of λ and ${}^{w}\Lambda^{\tau}$. Since the red triangle acts as a low-pass filter from the perspective of signal processing, its convolution with λ smooths out λ. Therefore, the second term of equation (53), $\frac{1}{4w^2}\int_{\tau-w}^{\tau+w}\big(\int_{x-w}^{x+w}\lambda dz\big)dx$, imposes smoothness constraint to λ.

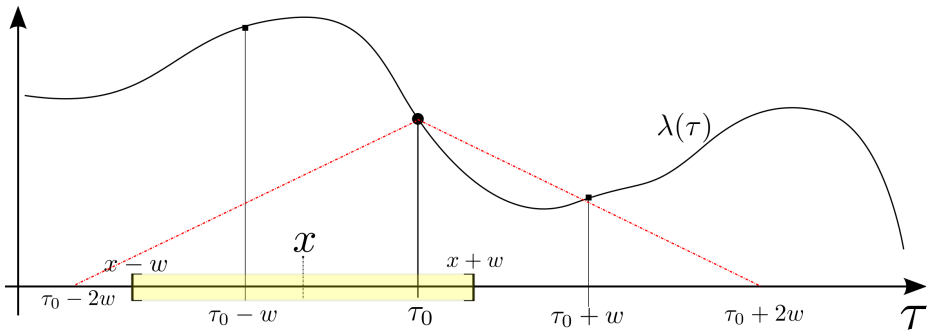
Based on the interpretation, when equation (53) is applied to λs with different local distributions around τ_o, as those shown in Figures 24(b) and 24(c), the resulting effects are illustrated by the blue dashed arrows. In Figure 24(b), the distribution of the camera parameter around τ_o is locally smooth, so the magnitude of equation (53) evaluated at τ_o is small. By contrast, when $\lambda(\tau)$ is locally sharp around τ_o, as shown in Figure 24(c), the magnitude of equation (53) evaluated at τ_o is large.

5.3 Experiments

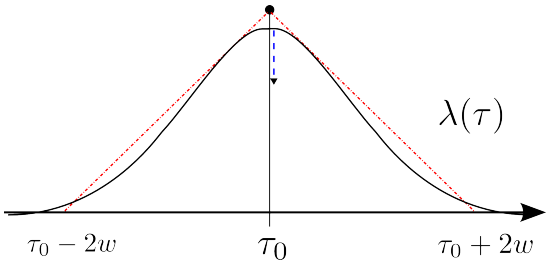
To validate the proposed algorithm, we conduct experiments on synthetic data developed using the techniques in chapter 4. We first generated smoothly corrupted camera parameters first (as those shown in Figure 25(a) or 25(b)), and then convert them to perturb the environment variables of virtual reality to simulate cameras perturbed by environmental factors.

The configuration of the synthetic data is illustrated in Figure 26. A deforming ocean surface is created in virtual reality and observed from the same positions as those shown in Figure 17 during a specified time interval. Artificial errors are added

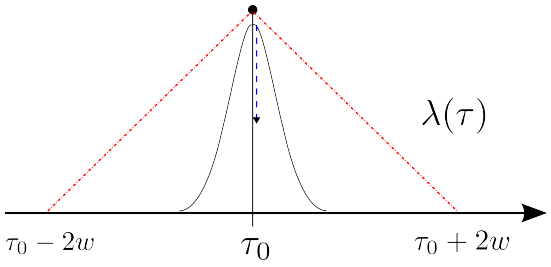
[2]A subscript 0 is added to τ to indicate the interval centered at τ_0.



(a) The L^2 inner product of the red triangle and λ within $[\tau_o - 2w, \tau_o + 2w]$ is equivalent to $\frac{1}{4w^2}\int_{\tau_o-w}^{\tau_o+w}\left(\int_{x-w}^{x+w}\lambda dz\right)dx$, the second term of equation (53). That is , $\frac{1}{4w^2}\int_{\tau_o-w}^{\tau_o+w}\left(\int_{x-w}^{x+w}\lambda dz\right)dx = \left\langle \lambda, {}^{w}\Lambda^{\tau_0}\right\rangle_{L^2(\tau_o-2w,\tau_o+2w)}$



(b) $\lambda(\tau)$ is locally smooth around τ_o.

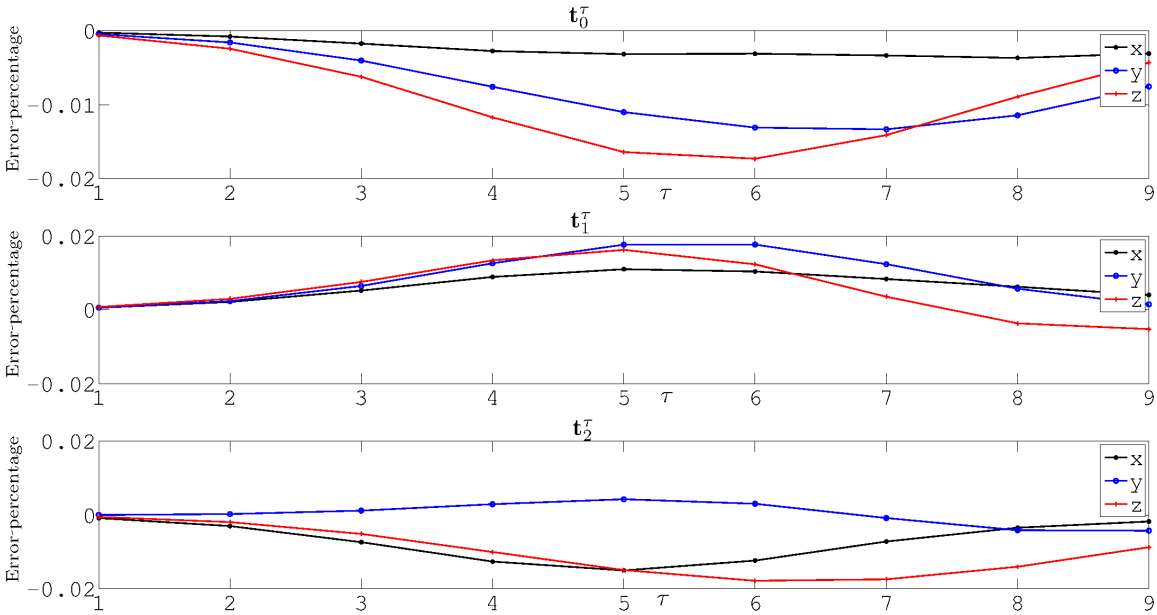


(c) $\lambda(\tau)$ is locally sharp around τ_o.

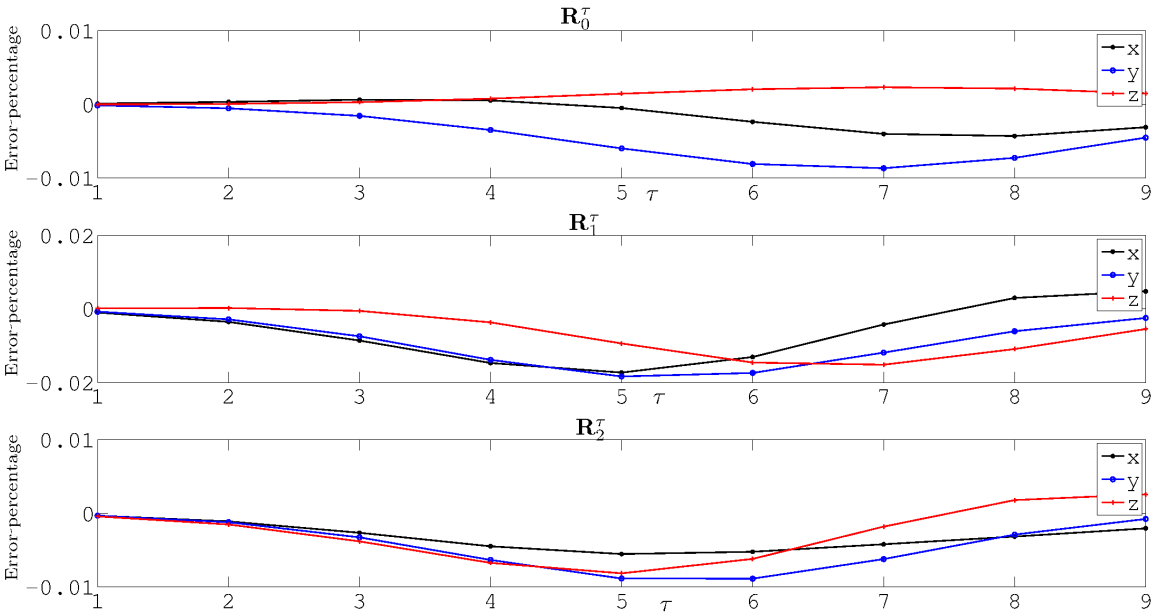
Figure 24: Figure 24(a) elucidates the smoothness effect that $\frac{1}{4w^2}\int_{\tau_o-w}^{\tau_o+w}\left(\int_{x-w}^{x+w}\lambda dz\right)dx$, the second term of equation (53), imposes on λ. Figures 24(b) and 24(c) illustrate how the entire equation (53) functions when it is applied to the shown situations.

to the true extrinsic camera parameters converted from the configuration of virtual scene developed using OpenGL. Since the purpose of this experiment is to explore the joint performance of the variational 4-D reconstruction and the corresponding calibration refinement algorithms under the smooth influence of environmental factors on the camera parameters, the artificial errors are deliberately smoothed within the observation interval (nine temporal samples), as shown in Figures 25(a) and 25(b).

We particularly take out the inputs and outputs at temporal samples 1, 5, and 9 for demonstration. The visual observations of the deforming ocean surface are listed in Figure 26(a), and the ground truth of the elevation maps produced through OpenGL at those time moments are listed in Figure 26(b). The 2-D dimensions of the surface are set to be $7,475.2$ mm, and the range of the height is configured to be

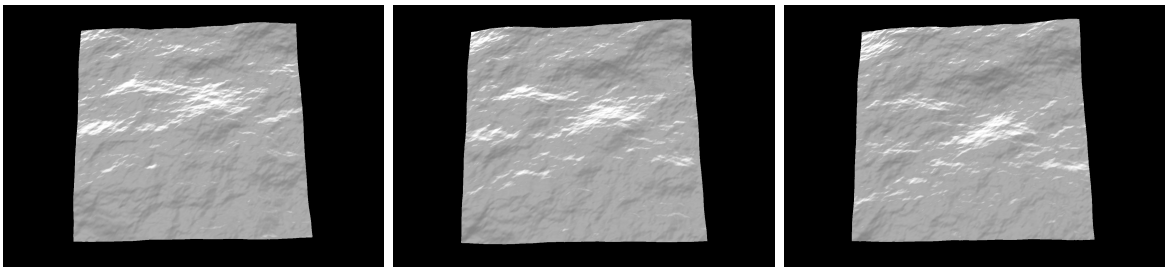


(a) Errors introduced to the translational part ($\mathbf{t}_i^\tau$) of the extrinsic parameters.

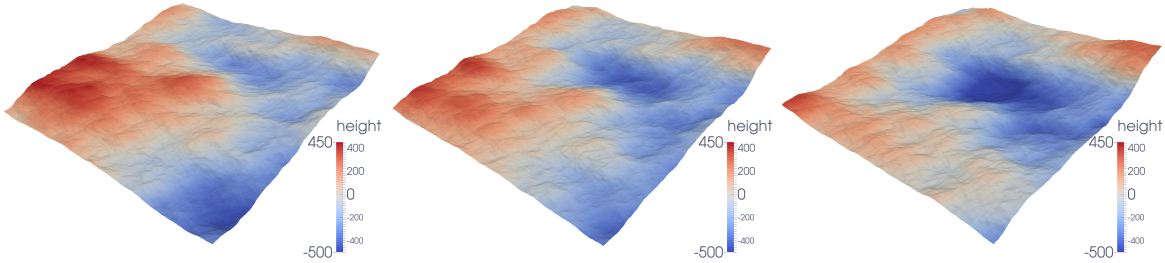


(b) Errors introduced to the rotational part (R_i^τ) of the extrinsic parameters.

Figure 25: Configuration of the artificial errors of the synthetic data. Artificial errors are temporally smooth and are added to corrupt the true camera parameters converted from OpenGL to simulate the perturbations imposed by natural factors on the cameras of a stereo computer vision rig. Note that the vertical axes indicate that errors are limited to be less than 2% of the true values.



(a) Snapshots taken at time samples 1, 5, and 9 by the middle camera in Figure 17.



(b) Elevation maps generated through OpenGL at time samples 1, 5, and 9.

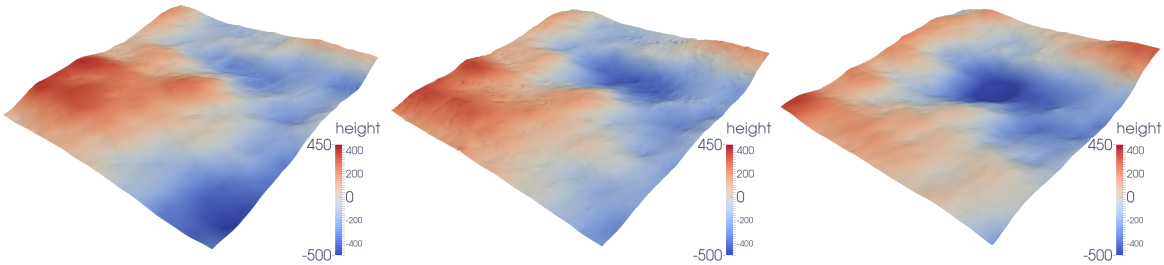
Figure 26: The synthetic data generated using OpenGL.

$[-500, 450]$ mm, as marked by the hue bars.

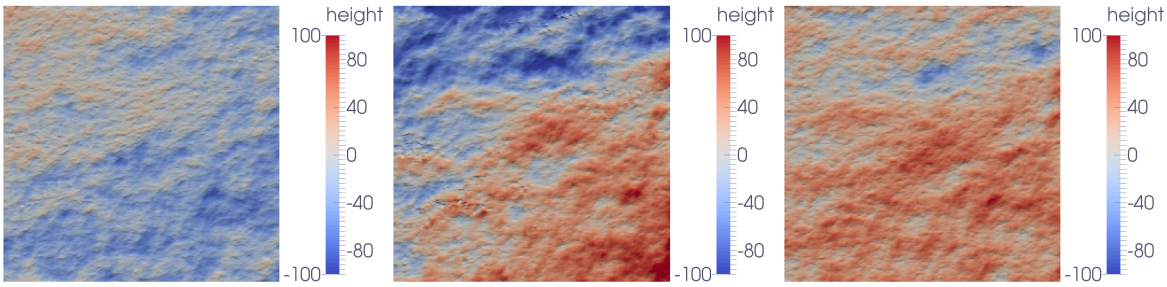
We initially set $\alpha = 5$, $\beta = 50$, $\gamma = 500$, and $w = 3$ for numerically solving equations (7) – (10) and (54). These parameters produce a rough reconstruction. Afterward, we continue to solve the same equations while α, β, γ, and w are gradually reduced to $\alpha = 0.2$, $\beta = 0.025$, $\gamma = 100$, and $w = 1$. Eventually, the elevation maps generated through this process are shown in Figure 27(a). In addition, the differences between Figures 26(b) and 27(a) are shown in Figure 27(b). All hue bars are given in the unit of millimeter.

An experiment in which the calibration refinement is not performed with the 4-D reconstruction is conducted and shown in Figure 28(a) for comparison. In this experiment, only equations (7)—(10) are solved; equation (54) is not used. The error maps between the outcomes of this experiment and the inputs are shown in Figure 28(b).

Because the artificial errors introduced at time 1 are small, the corresponding error maps (the first figures in Figures 27(b) and 28(b)) exhibit small differences

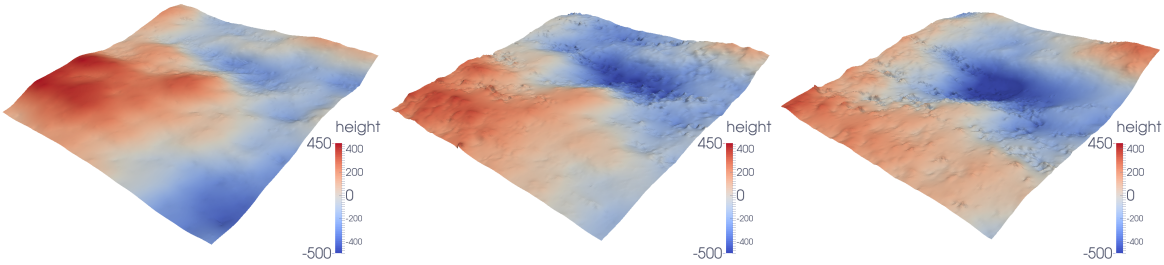


(a) Elevation maps (at time samples 1, 5, and 9) extracted from the space-time reconstruction generated through the joint operations of the 4-D reconstruction and the calibration refinement algorithms.

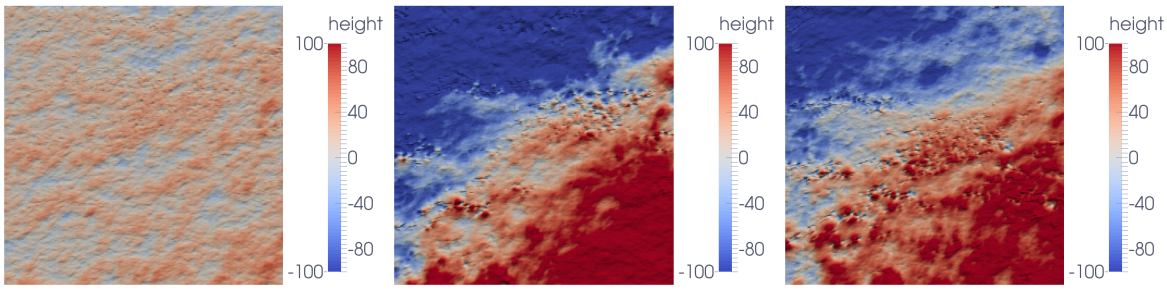


(b) Errors between Figures 26(b) and 27(a).

Figure 27: Output of the joint operations of the 4-D reconstruction and the calibration refinement algorithms.



(a) Elevation maps (at time samples 1, 5, and 9) extracted from the space-time reconstruction (without any calibration refinement).



(b) Errors between Figures 26(b) and 28(a).

Figure 28: Output of the space-time reconstruction algorithm in [21] applied to the synthetic data.

with respect to the ground truth shown in Figure 26(b). With increasing errors added to the subsequent moments, the error maps display more and more significant differences. In addition, the effect of the calibration refinement can be observed from the comparison between Figures 27(b) and 28(b).

In the case in which the reconstruction and the calibration refinement are jointly performed (as shown in Figure 27(b)), most of the points in the error maps at time samples 1, 5, and 9 fall within the ± 40 mm range. By contrast, when the reconstruction is executed alone, a very large portion of the error maps exhibit colors beyond the range of the hue bars.

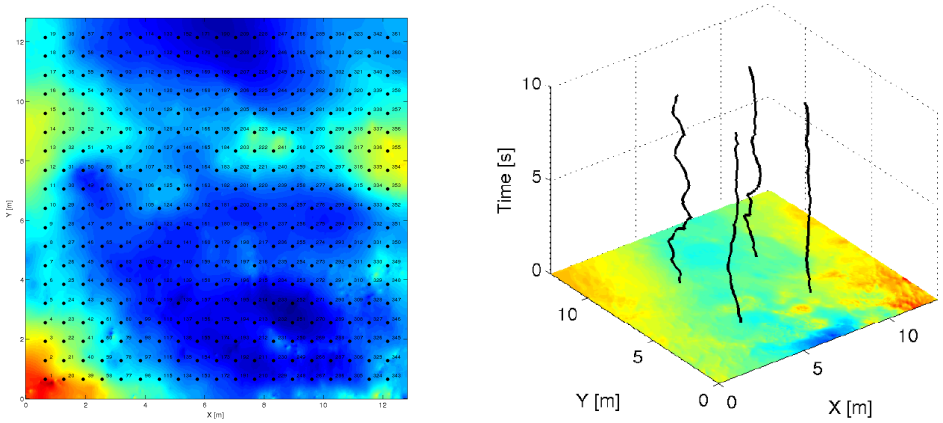
Note that even 1) with relatively large errors added to the camera parameters and 2) without the corresponding calibration refinement algorithm, the reconstruction at time samples 1, 5, and 9 in Figure 28(a) still roughly resemble the ground truth in Figure 26(b), which is mainly due to the temporal coherence imposed on the space-time reconstruction, as shown in equation (40).

CHAPTER VI

WAVE STATISTICS

The ultimate goal of this research project is to accurately reconstruct real ocean waves from snapshot pairs of two synchronous video sequences taken from offshore platforms, such as the Acqua Alta project in Figure 1. However, visually verifying a long sequence of reconstruction results for accuracy evaluations is impractical given that the 4-D measurements of ocean surfaces are unavailable. Luckily, we can rely on statistics as a validation tool. During decades of development of oceanography theories, many wave statistics in ocean engineering applications have been proposed and validated. For example, because of the the energy distribution of ocean waves, the elevation of an ocean surface behaves as a quasi-Gaussian random field. That is, height changes of a point on an ocean surface should be a Gaussian random process. Hence, we use how well our reconstructions obey this characteristic as a criterion of judging the accuracy of the reconstructed models.

In this chapter, wave statistics are applied to the space-time reconstruction of the dataset shown in Figure 3.6. As mentioned in section 3.6, the dimensions of the region of interest are 12.8 meters, and the distance between the ocean surface and the cameras is approximately 15.8 meters. The two synchronized videos, whose frame rates are 10, both contain 1025 snapshots. For each snapshot pair, we used a 513×513 spatial grid to represent the reconstructed elevation map. With 361 virtual probes evenly distributed on the 513×513 computation grid (illustrated in Figure 29(a)), the distance between each probe is 640 mm. Moreover, each extracted time sequence (illustrated in Figure 29(b)) has 1025 samples of elevation changes collected at the probe.



(a) 361 virtual probes are evenly placed to collect the data.

(b) Illustration of extracted time sequences

Figure 29: Analyses are applied to the height changes extracted from the space-time reconstruction.

Note that in the experiment, cameras are manually calibrated to the most optimal extent using the Camera Calibration Toolbox for MATLAB [6] before being used to film the ocean surface. Under the circumstances, not all wave statistics are expected to exhibit significant improvements after the calibration refinement method proposed in this thesis is added to jointly work with the variational ocean surface reconstruction algorithm in [21]. However, some types of statistics do exhibit the improvements carried out by the calibration refinement algorithm.

6.1 Statistical analyses

6.1.1 Average 1-D spectrum

According to [52], the mean 1-D spectrum[1]—the averaged spectrum of each time sequence—of real ocean waves should decay along a slope of -4 at frequencies near 1 Hz. The mean 1-D spectra of the space-time reconstructions generated through the algorithms in this thesis are shown in Figure 30. The slopes of both spectra near 1 Hz range between -4 and -5, agreeing with the physical prediction in [52]. Reference figures can be found in Figure 12 of [15] or Figure 18 of [4].

[1] A 1-D spectrum of a time sequence is obtained from its FFT

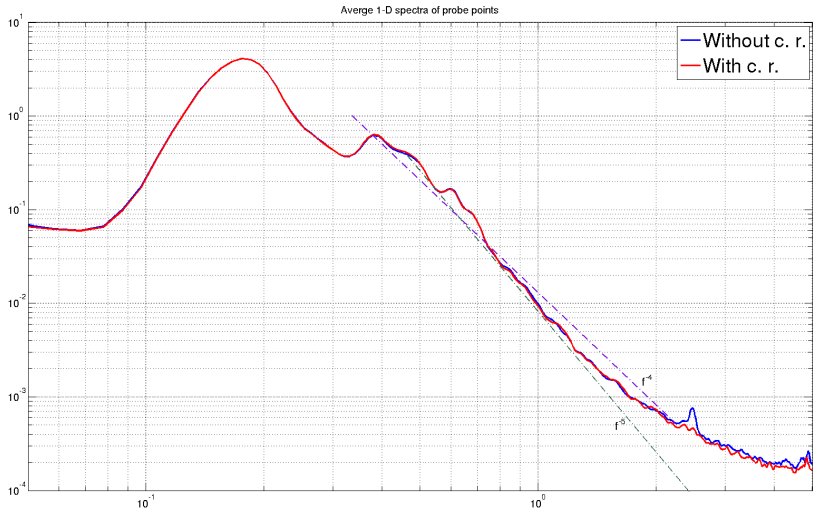


Figure 30: Average 1-D spectra. The bump shown on the blue curve at approximately 2.5 Hz disappears after the calibration refinement algorithm is added to jointly work with the variational ocean surface reconstruction algorithm.

Note that the two curves in Figure 30 almost coincide at the low frequency end. However, the "bump" shown on the blue curve at approximately 2.5 Hz vanishes from the spectrum after the calibration refinement algorithm is added to jointly work with the variational ocean surface reconstruction algorithm. The bump takes place on the high frequency end of the spectrum, and it is likely to arise from the perturbation introduced to the camera parameters during the videos were filmed. Even though the inception and the duration of the perturbation are unknown, the calibration refinement method proposed in this thesis can still successfully remove the effect of the perturbation.

6.1.2 The Euler characteristic

Originally developed to explore the topology properties of polyhedra, the Euler characteristic (EC) is applied to depict the randomness of density patterns in [50]. Adopted in [17] to explore the excursion set of a 2-D random field, the topological definition of EC is defined as

$$EC(h) = \#c.c. - \#holes, \tag{55}$$

in which $\#c.c.$ represents the number of connected components, and $\#holes$ indicates the number of "holes." Being observed from the top, when a plane with a height of h from the zero sea level is intersected with a reconstructed elevation map, the parts whose heights larger than h form holes (as illustrated in Figure 2 of [14]). The remaining regions not isolated by the holes are treated as individual connected components.[2] When the Euler characteristic is applied to a 2-D random field defined on a Cartesian grid, equation (55) can be rewritten as

$$EC(h) = \#P - (\#E_v + \#E_h) + \#F, \tag{56}$$

in which $\#P$ is the number of points with height above h, $\#E_v$ and $\#E_h$ are the numbers of the vertical edges and horizontal edges, respectively, and $\#F$ stands for the number of rectangular cells in the domain. Two definitions above would eventually lead to the same evaluation of the Euler characteristic for an given example.

The Euler characteristic is a practical tool of estimating the occurrence of extreme wave events because the number of 2-D upcrossings can be approximated by the Euler characteristic of excursion sets. When h is large, $EC(h)$ tends to zero; by contrast, when h is small (in negative value), $EC(h)$ tends to one.

Figure 31 shows the Euler characteristics of 1025 reconstructed elevation maps. Blue components in the figure correspond to the results generated through the variational reconstruction algorithm alone; red components are related to the elevation maps generated through the joint operations of the variational reconstruction and calibration refinement algorithms. Vertical bars represent the range of $[EC - \sigma(EC),\ EC - \sigma(EC)]$, in which σ represents the corresponding standard deviation. Compared to the curve of the expected EC with boundary corrections shown in Figure 15 in [16], the curve with calibration refinement is more similar to the Euler characteristic of a Gaussian random field.

[2] If a hollow region has any connection to the boundary, it should not be treated as a hole by definition. A hole should be isolated from the boundaries.

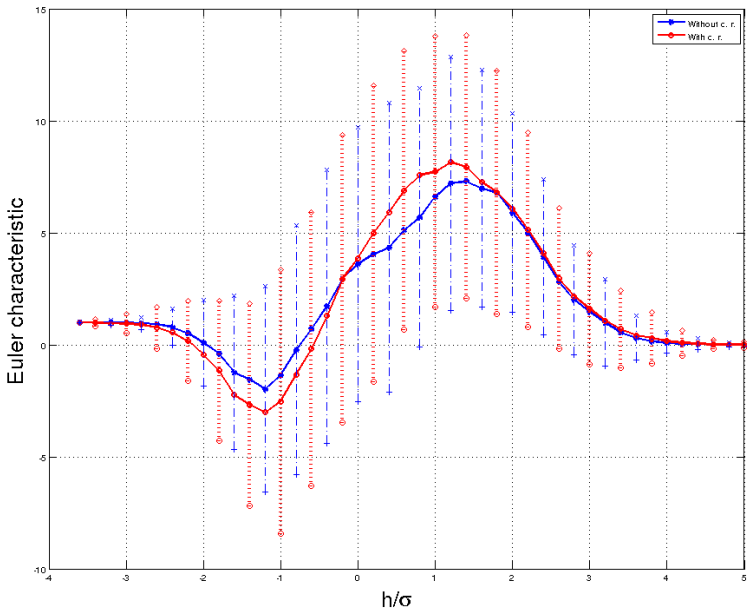


Figure 31: The Euler characteristics of the reconstructed elevation maps are shown as the two curves in the middle. Vertical bars represent the range of $[EC-\sigma(EC),\ EC-\sigma(EC)]$, in which σ represents the corresponding standard deviation.

6.1.3 Wave height exceedance probability

The Boccotti asymptotic form, defined as

$$P(h > H) = c\ \exp\Big(-\frac{H^2}{4\sigma^2(1+\psi^*)}\Big), \tag{57}$$

is given in [5] to predict empirical wave height distributions. In equation (57), c and ψ^*[3] both depend on the first minimum of the wave covariance. In our case, $c \approx 1$ and $\psi^* \approx 0.4526$.

The wave height exceedance probability can also be predicted by the following deduction. If both h and H are large numbers, the expected number of H-upcrossings can be approximated by $EX(H)$, so the wave height exceedance probability is

$$P(h > H) = \frac{EX(H)}{\bar{T}} = \exp\Big(\frac{-H^2}{2m_0}\Big). \tag{58}$$

[3]The autocovariance of a time series $\eta(t)$ is defined as $\psi(T) = \langle \eta(t)\eta(t+T)\rangle$, and $\psi^* = |\psi(T^*)|$, in which $T^*(T^* > 0)$ is the value at which the first local minimum of ψ takes place.

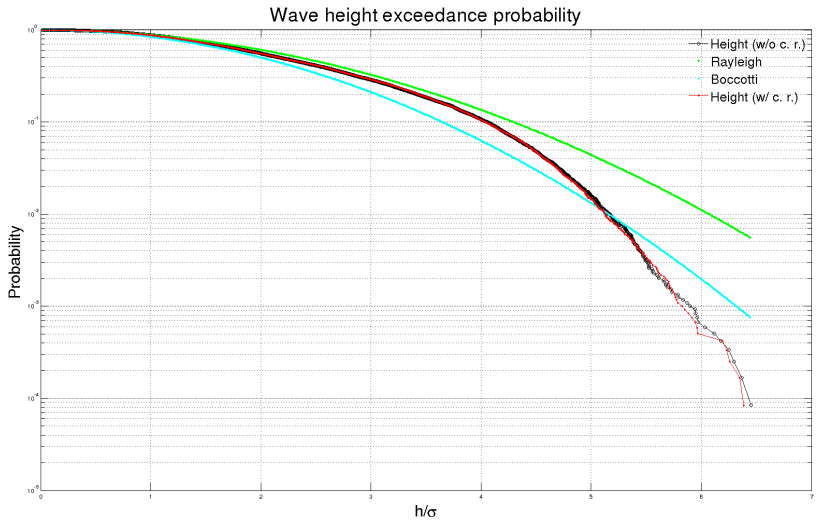


Figure 32: Wave height exceedance probability

In this equation, $EX(.)$ is given by Rice's formula as

$$EX(h) = \frac{\tau}{2\pi}\sqrt{\frac{m_2}{m_0}}\exp\left(\frac{-h^2}{2m_0}\right),$$

in which $m_j = \int_0^\infty \omega^j S(\omega) d\omega$ and ω is the angular frequency. $\bar{T}$, the mean wave period, is $\bar{T} = \frac{\tau}{EX(0)} = 2\pi\sqrt{\frac{m_0}{m_2}}$. Equation (58) can be written as

$$P(h > H) = \exp\left(\frac{-H^2}{8m_0}\right) \tag{59}$$

if crests and troughs are symmetric (Gaussian sea states). As a result, equation (59) forms a Rayleigh distribution.

According to the theories, the empirical exceedance probability should be bounded between Rayleigh (equation (59)) and Boccotti (equation (57)) distributions, and both curves in Figure 32 do agree with the prediction.

6.1.4 Empirical PDF

To verify whether the elevation maps generated from our algorithms follow the characteristics of a quasi-Gaussian random field, we compare the empirical probability

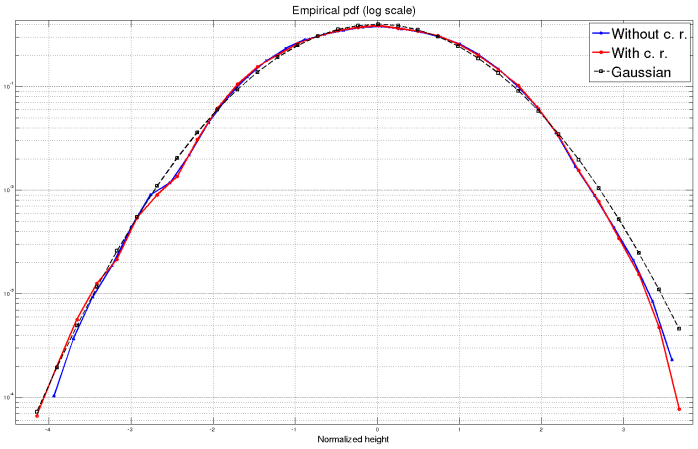


Figure 33: Empirical PDFs

density function (PDF) of the series of elevation maps to a standard Gaussian distribution (with a zero mean and a unit standard deviation). We stacked each normalized time sequence[4] to yield a composite one, and the empirical PDF is defined as the histogram of this composite sequence. According to the theory in [46], the empirical PDFs of the reconstructed elevation maps (with or without the calibration refinement algorithm) should be close to a standard Gaussian distribution. This phenomenon can be observed in Figure 33.

6.1.5 Miscellaneous: wave skewness and kurtosis

Skewness and kurtosis are statistics for the measurements of the asymmetry and peakedness of a distribution, respectively. Since the magnitude of mean wave crests is usually larger than that of wave troughs, the average skewness of real ocean surface ranges between 0 and 0.3, and the average kurtosis is around 3. The results of the statistics obtained from different cases are shown in Table 7. The average skewness and kurtosis of the 1025 elevation maps generated from the joint operations of the variational and calibration refinement algorithms are shown on the last column; in this case, both statistics agree with theoretical results. By contrast, the average

[4]Normalizing a time sequence means that each sample in the sequence is divided by their standard deviation after the average component is subtracted.

Table 7: Kurtosis and skewness obtained from the reconstructions. The boldfaced values are the statistics that agree with theoretical estimations.

	without c. r.	with c. r.
skewness	0.3301	**0.1938**
kurtosis	**2.849**	**2.891**

skewness of the elevation maps generated from variational reconstruction alone does not fall in the predicted range.

CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1 Conclusion

We have designed a novel camera calibration refinement algorithm specifically for the variational wave acquisition stereo system (VWASS), which performs the variational spatio-temporal reconstruction of ocean waves. In this context, the recovery of the spatio-temporal behavior of ocean waves and the refinement of the camera parameters are jointly formulated into the minimizers of an optimization framework. The primary purpose of coupling calibration refinement with the VWASS is to improve the reconstruction accuracy and robustness of the VWASS. As illustrated in previous chapters, the work presented in this thesis is indeed a promising step toward robust stereoscopic methods of determining the 4-D ocean surface features from video recordings, which may ultimately offer the possibility of overcoming many of the uncertainties related to conventional wave measuring devices. Therefore, the topic we addressed—reducing the errors associated with camera parameter deviations—is particularly important in ocean engineering since such measurement systems would find applications on offshore and gas facilities in which camera motions are unavoidable.

Because of the complexity of our algorithms, computation efficiency becomes an important issue for the progress of the research. As a result, the implementation of the algorithms was particularly designed so that they can be accelerated by high performance techniques, including hardware and software. Meanwhile, numerical schemes are prudently chosen to strike a balance between the calculation cost and execution efficiency. The efforts for boosting the computation speed as well as maintaining the stability of the algorithm will not only expedite relative scientific research based on

the current framework but also increase the possibility that the VWASS is adopted in industries, such as oil companies.

Experiments are conducted to validate the correctness and effectiveness of the proposed algorithms and the selected numerical schemes. We use known statistics of ocean surfaces to validate the outcome of the joint performance of the variational 4-D reconstruction and the camera calibration refinement applied to a patch of real ocean surface whose dimensions are 12.8 meters in width and in length. Furthermore, considering that the ground truth of wave heights is difficult to be obtained, we particularly developed synthetic data for validating the algorithm performance. The development of the synthetic ocean surfaces is an extremely important achievement for this project for the following reasons:

- The real data have been collected through two synchronized cameras by our research partners working on an offshore platform at the Black Sea. Coordinating with them to increase the number of installed cameras as well as to precisely calibrate them is a long and tedious process. However, the usage of synthetic data substantially shortens the time for acquiring data.

- We anticipate that the performance and stability of our algorithms can be substantially improved with the number of installed cameras increased. However, increasing the number of cameras in the system will inevitably introduce more calibration errors (because of manual operations or measurement deviations) to the parameter estimations of the stereo rig, conflicting with our original goal. With the help of synthetic data, we could not only arbitrarily increase the number of cameras without introducing any unfavorable calibration errors, but also deliberately corrupt specific camera parameters for validation, which benefit the designs of new algorithms in the future.

- Statistics are sometimes not effective for validating the reconstructions. For

instance, two reconstucted elevation maps with different averages may have the same standard deviation. In this case, the statistics without considering the means fail to verify the correctness of the reconstructions. By contrast, with the direct comparison between the inputs (synthetic data created via computer graphics) and outputs (the reconstruction generated through our algorithms), even a user without enough statistics training can visually judge the correctness of the reconstructions.

7.2 Future work

Because of reflection and refraction of water surfaces, a physical point on a water surface may exhibit different colors to observers at different positions, which is a phenomenon called specularity. This natural phenomenon conflicts with the fundamental assumption of equation (5)[1] and consequently undermines the reconstruction quality of the VWASS, in which only two cameras are used to record an ocean surface (see the pictures in Figure 1). A possible remedy for this undesirable situation is to force the rank of the radiance tensor to be less than or equal to two [27, 28], implicating that the number of cameras filming the scene should be increased up to at least three. However, as previously mentioned, increasing the number of cameras introduces more potential calibration errors. Therefore, attention should particularly be paid to this situation.

Moreover, the VWASS is currently limited to reconstruct ocean waves with small amplitudes. In this case, the visual fields of cameras can cover all points in the region of interest. When waves are high, some portions of the ocean surface may be occluded by other parts and, consequently, invisible to some camera. To reconstruct this type of waves, we should increase the number of cameras and "coordinate" the cameras to "collaborate" on the reconstruction: The reconstruction is an assembled work of

[1]Recall that a universal radiance function, f, is used to represent the superficial pattern on the reconstructed region. This type of surface is called a Lambertian surface.

several pieces of reconstructions generated by different sets of cameras. This concept is similar to the operation in [35]. However, because the outputs of the VWASS are piece-wise smooth functions, the registration of these 3-D graphs requires advanced differential geometry concepts to achieve.

Furthermore, since the multigrid method [8] is used to numerically solve the PDEs with astonishingly high convergence rate, it offers an alternative for solving the differential equations related to the calibration refinement algorithms proposed in previous chapters, which currently rely on the gradient descent method for locating local minima. In addition, advanced computer graphics techniques, such as shadowing and lighting, can be utilized to produce synthetic ocean surfaces with natural appearances. In addition, general-purpose computing on graphics processing units (GPGPUs) can be utilized to substitute for the current high performance computing facilities—clusters. After all, clusters are unlikely to be deployed on an offshore platform for on-site ocean surface reconstructions.

APPENDIX A

GRADIENT VECTORS FOR MINIMIZING THE CONSTRAINED DATA FIDELITY TERM

Denote the camera parameter under discussion λ.[1] By the chain rule of calculus, the derivative of $E_{data,i}$ (equation (13)) with respect to λ is

$$\frac{\partial E_{data,i}}{\partial \lambda} = \left\{ \frac{\partial(\frac{1}{A_{R_i}})}{\partial \lambda} \int_{R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}} \right.$$
$$\left. + (\frac{1}{A_{R_i}}) \frac{\partial(\int_{R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}})}{\partial \lambda} \right\}. \tag{60}$$

Note that in the second term of equation (60),

$$\frac{\partial(\int_{R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}})}{\partial \lambda} \neq \int_{R_i} \frac{\partial}{\partial \lambda} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}}$$

because the region of integration, R_i, also depends on λ. To tackle this problem, we introduce Reynolds' transport theorem. Recall that Reynolds' transport theorem in a 2-D case states that

$$\frac{d}{dt} \int_{R_i} \boldsymbol{F}(\boldsymbol{x}, t) d\boldsymbol{x} = \int_{R_i} \frac{\partial}{\partial t} \boldsymbol{F}(\boldsymbol{x}, t) d\boldsymbol{x} + \int_{\partial R_i} \boldsymbol{F}(\boldsymbol{x}, t) \big\langle \frac{\partial \boldsymbol{x}}{\partial t}, \hat{n} \big\rangle d\hat{s},$$

where R_i is a 2-D closed region dependent on variable t, ∂R_i is the boundary of R_i, $\boldsymbol{F}(\boldsymbol{x}, t)$ is a vector field defined within R_i and on ∂R_i, $d\boldsymbol{x}$ is the area element of R_i, $\hat{n}$ is the outward unit normal on ∂R_i, and $d\hat{s}$ is the arc length element of ∂R_i. We then simplify $\frac{\partial E_{data,i}}{\partial \lambda}$ by applying this theorem to the second term of (60), yielding

[1] λ represents a three-by-one or two-by-one vector.

$$\frac{\partial E_{data,i}}{\partial \lambda} = \Bigg\{ \frac{\partial (A_{R_i}^{-1})}{\partial \lambda} \int_{R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}} \tag{61}$$

$$+ \ (\frac{1}{A_{R_i}}) \Big\{ \int_{R_i} \frac{\partial}{\partial \lambda} \Big(\Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 \Big) d\hat{\mathbf{x}} \tag{62}$$

$$+ \ (\frac{1}{A_{R_i}}) \int_{\partial R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 \Big\langle \hat{\mathsf{n}}, \frac{\partial \hat{\mathsf{c}}}{\partial \lambda} \Big\rangle d\hat{\mathsf{s}} \Bigg\}. \tag{63}$$

The simplification of equation (61) requires $\frac{\partial A_{R_i}}{\partial \lambda}$, which needs some tricks to be obtained. Recall that the divergence theorem states that

$$\int_{\partial R_i} \langle \boldsymbol{F}, \hat{n} \rangle \mathrm{d}\hat{s} = \int_{R_i} (\nabla \cdot \boldsymbol{F}) d\boldsymbol{x}. \tag{64}$$

In the 2-D case, $\boldsymbol{F}$ can be arbitrarily chosen as $[x, y]^{\mathsf{T}}$, then equation (64) becomes

$$\int_{\partial R_i} \langle \begin{bmatrix} x \\ y \end{bmatrix}, \hat{n} \rangle \mathrm{d}\hat{s} = \int_{\partial R_i} \langle \hat{c}, \hat{n} \rangle \mathrm{d}\hat{s} = \int_{R_i} (\nabla \cdot \begin{bmatrix} x \\ y \end{bmatrix}) d\boldsymbol{x} = 2 \int_{R_i} d\boldsymbol{x} = 2A_{R_i},$$

so

$$A_{R_i} = \frac{1}{2} \int_{\partial R_i} \langle \hat{\mathsf{c}}, \hat{\mathsf{n}} \rangle \mathrm{d}\hat{\mathsf{s}}.$$

In addition, $\frac{\partial A_{R_i}}{\partial \lambda}$ can be derived through Reynolds' transport theorem again, so

$$\frac{\partial A_{R_i}}{\partial \lambda} = \frac{\partial}{\partial \lambda} \int_{R_i} d\boldsymbol{x} = \cancel{\int_{R_i} \frac{\partial (1)}{\partial \lambda} \mathrm{d}\boldsymbol{x}} + \int_{\partial R_i} \langle \frac{\partial \hat{\mathsf{c}}}{\partial \lambda}, \hat{\mathsf{n}} \rangle \mathrm{d}\hat{\mathsf{s}} = \int_{\partial R_i} \langle \frac{\partial \hat{\mathsf{c}}}{\partial \lambda}, \hat{\mathsf{n}} \rangle \mathrm{d}\hat{\mathsf{s}}.$$

As a result, equation (61)

$$= \ -A_{R_i}^{-2} \int_{R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}} \frac{\partial A_{R_i}}{\partial \lambda}$$

$$= \ -A_{R_i}^{-2} \int_{R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 d\hat{\mathbf{x}} \int_{R_i} \langle \hat{\mathsf{n}}, \frac{\partial \hat{\mathsf{c}}}{\partial \lambda} \rangle \mathrm{d}\hat{\mathsf{s}}. \tag{65}$$

Term (62) can be further simplified as follows:

$$\frac{2}{A_{R_i}} \int_{R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big] \Bigg(\cancel{\frac{\partial I_i\big(\hat{\mathbf{x}}\big)}{\partial \lambda}} - w_i \frac{\partial f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big)}{\partial \lambda} \Bigg) d\hat{\mathbf{x}}$$

$$= \ \frac{2}{A_{R_i}} \int_{R_i} w_i \Big[w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i - I_i\big(\hat{\mathbf{x}}\big) \Big] \Bigg(\frac{\partial f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big)}{\partial \lambda} \Bigg) d\hat{\mathbf{x}}$$

$$= \ \frac{2}{A_{R_i}} \int_{R_i} w_i \Big[w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i - I_i\big(\hat{\mathbf{x}}\big) \Big] \Big\langle \nabla_{\mathfrak{s}} f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big), \frac{\partial \pi_i^{-1}(\hat{\mathbf{x}})}{\partial \lambda} \Big\rangle d\hat{\mathbf{x}}, \tag{66}$$

in which $\nabla_{\mathfrak{s}} f$ stands for $\left(\frac{\partial f}{\partial \mathbf{X}}\right)^{\mathsf{T}}$, which is $(f_u, f_v, 0)^{\mathsf{T}}$. In equation (63), unit outward normal $\hat{\mathfrak{n}}$ can be represented by the corresponding unit tangent $\hat{\mathbf{t}}$ rotated by 90- or 270-degree. This rotation is represented as a two-by-two matrix Q. Therefore, Term (63) becomes

$$(\frac{1}{A_{R_i}}) \int_{\partial R_i} \Big[I_i\big(\hat{\mathbf{x}}\big) - \big(w_i f\big(\pi_i^{-1}(\hat{\mathbf{x}})\big) + \gamma_i\big) \Big]^2 \left\langle Q\hat{\mathbf{t}}, \frac{\partial \hat{\mathfrak{c}}}{\partial \lambda} \right\rangle d\hat{\mathfrak{s}}. \tag{67}$$

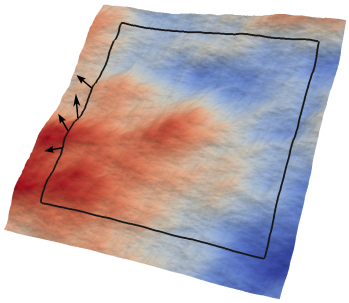
The sum of equations (65)–(67) are not yet the ultimate analytical representation of $\frac{\partial E_{data,i}}{\partial \lambda}$ since π_i^{-1} has no analytical form. However, π_i does, so we should manage to convert $\frac{\partial \pi_i^{-1}(\hat{\mathbf{x}})}{\partial \lambda}$ to $\frac{\partial \pi_i(\hat{\mathbf{x}})}{\partial \lambda}$ to get a further simplification for $\frac{\partial E_{data,i}}{\partial \lambda}$. By using a change of variables, we lift the region of the integral from the image domain to world coordinate system U, yielding

$$\begin{aligned}
\frac{\partial E_{data,i}}{\partial \lambda} \;=\; & \left\{ \frac{2}{A_{R_i}} \int_U w_i \Big[w_i f\big(\mathbf{X}\big) + \gamma_i - I_i\big(\pi_i(\mathbf{X})\big) \Big] \left\langle \nabla_{\mathfrak{s}} f\big(\mathbf{X}\big), \frac{\partial (\mathbf{X})}{\partial \lambda} \right\rangle J_i d\boldsymbol{u} \right. \\
& - A_{R_i}^{-2} \int_U \Big[I_i\big(\pi_i(\mathbf{X})\big) - \big(w_i f\big(\mathbf{X}\big) + \gamma_i\big) \Big]^2 J_i d\boldsymbol{u} \int_{\partial U} \left\langle Q \frac{\partial \big(\pi_i(\mathbf{X})\big)}{\partial s}, \frac{\partial \big(\pi_i(\mathbf{X})\big)}{\partial \lambda} \right\rangle ds \\
& + \left. \frac{1}{A_{R_i}} \int_{\partial U} \Big[I_i\big(\pi_i(\mathbf{X})\big) - \big(w_i f\big(\mathbf{X}\big) + \gamma_i\big) \Big]^2 \left\langle Q \frac{\partial \big(\pi_i(\mathbf{X})\big)}{\partial s}, \frac{\partial \big(\pi_i(\mathbf{X})\big)}{\partial \lambda} \right\rangle ds \right\}.
\end{aligned}$$

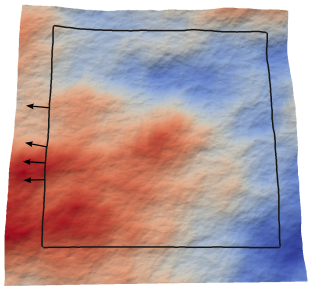
In the equations above, ∂U is the boundary of U, and U is the spatial domain defined in the illustration of Figure 4.

The complicated expressions above cannot be further analytically simplified because the numerical implementation of $\hat{\mathfrak{n}}$ (2-D outward normals on the reprojection of 3-D artificially-defined boundaries, shown as the arrows in Figure 34(b)) is necessary. Since the 3-D boundaries of the reconstructed region (region of interest, black contours in Figure 34(a)) are artificially determined and, consequently, are unlikely to coincide with the natural occluding boundaries, $\hat{\mathfrak{n}}$ cannot be analytically obtained through the reprojection of its corresponding surface normal (denoted by $\boldsymbol{N}$, shown in Figure 34(a)). Hence, we numerically implemented $\hat{\mathfrak{n}}$ using $Q\frac{\partial \big(\pi_i(\mathbf{X})\big)}{\partial s}$.

Different camera parameters have different analytical forms for $\frac{\partial (\mathbf{X})}{\partial \lambda}$ and $\frac{\partial \big(\pi_i(\mathbf{X})\big)}{\partial \lambda}$, so we derived them as follows:



(a) Surface normals ($\boldsymbol{N}$): 3-D vectors on the artificially-determined boundaries perpendicular to the elevation map.



(b) Contour normals ($\hat{\mathsf{n}}$ in equation (63)): 2-D vectors perpendicular to the reprojection of 3-D boundaries.

Figure 34: Normals (black arrows) on the boundaries of the reconstructed region (black contours). Figures 34(a) and 34(b) demonstrate that a 2-D normal cannot be analytically obtained by the reprojection of the corresponding 3-D normal.

A.1 $\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda}$

According to Figure 3 and equation (2),

$$\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda} = \frac{\partial\left(\hat{\mathbf{x}}\right)}{\partial\lambda} = \frac{\partial}{\partial\lambda}\left(\begin{bmatrix} L_x & a & \zeta_0 \\ 0 & L_y & \eta_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \tilde{\mathsf{X}}_i \\ \tilde{\mathsf{Y}}_i \\ \tilde{\mathsf{Z}}_i \end{bmatrix}\right)_h = \frac{\partial}{\partial\lambda}\begin{bmatrix} L_x\frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i} + a\frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \zeta_0 \\ L_y\frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \eta_0 \end{bmatrix}, \quad (68)$$

so when λ stands for the intrinsic camera parameters, we have

$$\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda} = \begin{cases} \frac{1}{\tilde{\mathsf{Z}}_i}\begin{bmatrix} \tilde{\mathsf{X}}_i & 0 \\ 0 & \tilde{\mathsf{Y}}_i \end{bmatrix} & \text{if } \lambda = \left(L_x, L_y\right)^{\mathsf{T}} \\ \mathbf{I} & \text{if } \lambda = \left(\zeta_0, \eta_0\right)^{\mathsf{T}} \end{cases}. \quad (69)$$

If $\lambda = \mathbf{t}_i$, the three-by-one translation vector of the extrinsic camera parameters, $\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda}$ becomes

$$\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda} = \frac{\partial\begin{bmatrix}(L_x\frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i} + a\frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i}) + \zeta_0 \\ L_y\frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \eta_0\end{bmatrix}}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\tilde{\mathbf{X}}_i}\frac{\partial\tilde{\mathbf{X}}_i}{\partial\lambda} = \underbrace{\begin{bmatrix} L_x & a \\ 0 & L_y \end{bmatrix}\begin{bmatrix} \frac{1}{\tilde{\mathsf{Z}}_i} & 0 & -\frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i^2} \\ 0 & \frac{1}{\tilde{\mathsf{Z}}_i} & -\frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i^2} \end{bmatrix}}_{\equiv\boldsymbol{\phi}}\frac{\partial\tilde{\mathbf{X}}_i}{\partial\lambda} = \boldsymbol{\phi}\mathbf{I} = \boldsymbol{\phi}.$$

Note that the multiplication result of matrices $\begin{bmatrix} L_x & a \\ 0 & L_y \end{bmatrix}$ and $\begin{bmatrix} \frac{1}{\tilde{\mathsf{Z}}_i} & 0 & -\frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i^2} \\ 0 & \frac{1}{\tilde{\mathsf{Z}}_i} & -\frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i^2} \end{bmatrix}$ are particularly denoted by $\boldsymbol{\phi}$ for succinctness.

The only type of remaining camera parameters that have not been discussed is R_i, a three-by-three rotation matrix. R_i has three degrees of freedom. According to Rodrigues' rotation formula, a three-by-three rotation matrix, R_i, is associated with its rotation vector, $\vec{\omega}_i$, by

$$\mathsf{R}_i = e^{[\vec{\omega}_i]_{\mathbf{x}}},$$

in which $\vec{\omega}_i \equiv \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^{\mathsf{T}}$ and $[\vec{\omega}_i]_{\mathbf{x}} \equiv \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$. When λ represents $\vec{\omega}_i$,

$$\frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial\lambda} = \boldsymbol{\phi}\frac{\partial\tilde{\mathbf{X}}_i}{\partial\lambda} = \boldsymbol{\phi}\frac{\partial(\mathsf{R}_i\mathbf{X})}{\partial\lambda} = -\boldsymbol{\phi}\mathsf{R}_i[\mathbf{X}]_{\mathbf{x}}\frac{\Big(\vec{\omega}_i\vec{\omega}_i^{\mathsf{T}} + (\mathsf{R}_i^{\mathsf{T}} - \mathbf{I})[\vec{\omega}_i]_{\mathbf{x}}\Big)}{\|\vec{\omega}_i\|^2}.$$

The derivations of obtaining equation $\frac{\partial(\mathsf{R}_i\mathbf{X})}{\partial\lambda} = -\mathsf{R}_i[\mathbf{X}]_{\mathbf{x}}\frac{\Big(\vec{\omega}_i\vec{\omega}_i^{\mathsf{T}}+(\mathsf{R}_i^{\mathsf{T}}-\mathbf{I})[\vec{\omega}_i]_{\mathbf{x}}\Big)}{\|\vec{\omega}_i\|^2}$ is explored in [20].

A.2 $\frac{\partial(\mathbf{X})}{\partial\lambda}$

Assume that all camera parameters are accurate. In this case, point $\mathbf{X}_o$ in the world coordinate system is projected onto point $\hat{\mathbf{x}}_o$ on the i^{th} image plane via π_i. However, in real situations, because of camera calibration errors, $\mathbf{X}_o + \Delta\mathbf{X}$ (as opposed to $\mathbf{X}_o$) is projected onto $\hat{\mathbf{x}}_o$, which indicates that $\Delta\mathbf{X}$ nullifies the errors of the camera parameters so that $\frac{\partial\hat{\mathbf{x}}}{\partial\lambda}\Big|_{\hat{\mathbf{x}}=\hat{\mathbf{x}}_o} = 0$. Therefore, we can derive $\frac{\partial(\mathbf{X})}{\partial\lambda}$ through the following equation:

$$\frac{\partial\hat{\mathbf{x}}}{\partial\lambda} = 0 = \frac{\partial\pi_i(\mathbf{X},\pi^*)}{\partial\lambda} = \underbrace{\frac{\partial\pi_i}{\partial\mathbf{X}}\frac{\partial\mathbf{X}}{\partial\lambda}}_{E1} + \underbrace{\frac{\partial\pi_i}{\partial\pi^*}\frac{\partial\pi^*}{\partial\lambda}}_{E2}, \tag{70}$$

where π^* is any intermediate transformation of π_i. For instance, it can be the linear transformation that maps $\mathbf{X}$ to $\tilde{\mathbf{X}}_i$; in this case, π^* is the coordinate transformation

of equation (1). In equation (70), $E1$ is $\boldsymbol{\phi}\mathsf{R}_i\frac{\partial \mathbf{X}}{\partial \lambda}$,[2] but $E2$ is variable-dependent. We discuss them as follows:

If $\lambda = \mathbf{t}_i$, then

$$E2 = \frac{\partial \pi_i}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \tilde{\mathbf{X}}_i} \frac{\partial \tilde{\mathbf{X}}_i}{\partial \lambda} = \boldsymbol{\phi} \frac{\partial \tilde{\mathbf{X}}_i}{\partial \lambda}.$$

Now that $E1 + E2 = 0$, we get

$$\boldsymbol{\phi}\mathsf{R}_i \frac{\partial \mathbf{X}}{\partial \lambda} = -\boldsymbol{\phi} \frac{\partial \tilde{\mathbf{X}}_i}{\partial \lambda}. \tag{71}$$

Recall that $\boldsymbol{\phi}$ is defined as

$$\boldsymbol{\phi} = \begin{bmatrix} L_x & a \\ 0 & L_y \end{bmatrix} \begin{bmatrix} \frac{1}{\tilde{\mathsf{Z}}_i} & 0 & -\frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i^2} \\ 0 & \frac{1}{\tilde{\mathsf{Z}}_i} & -\frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i^2} \end{bmatrix},$$

so the rank of $\boldsymbol{\phi}$ is two. As such, equation (71) has no unique solution. However, since $\frac{\partial \mathbf{X}}{\partial \lambda}$ is on the tangent plane of surface $S(\boldsymbol{u})$, $\frac{\partial \mathbf{X}}{\partial \lambda}$ is perpendicular to the normal on $S(\boldsymbol{u})$. Therefore,

$$\left\langle \frac{\partial \mathbf{X}}{\partial \lambda}, \boldsymbol{N} \right\rangle = 0 = \left\langle \frac{\partial \mathbf{X}}{\partial \lambda}, \mathsf{R}_i^{-1} \boldsymbol{N}_i \right\rangle = \left\langle \mathsf{R}_i \frac{\partial \mathbf{X}}{\partial \lambda}, \boldsymbol{N}_i \right\rangle, \tag{72}$$

where $\boldsymbol{N}_i$ is the observation of $\boldsymbol{N}$ in the i^{th} camera coordinate system.

Now, we define the following symbols:

$$\Phi \equiv \begin{bmatrix} \boldsymbol{\phi} \\ \mathbf{N}_i^{\mathsf{T}} \end{bmatrix} \equiv \begin{bmatrix} \vec{\phi_1}^{\mathsf{T}} \\ \vec{\phi_2}^{\mathsf{T}} \\ \boldsymbol{N}_i^{\mathsf{T}} \end{bmatrix} \equiv \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \boldsymbol{N}_i.x & \boldsymbol{N}_i.y & \boldsymbol{N}_i.z \end{bmatrix},$$

then we combine equations (71) and (72), yielding

$$\begin{bmatrix} \boldsymbol{\phi} \\ \mathbf{N}_i^{\mathsf{T}} \end{bmatrix} \mathsf{R}_i \frac{\partial \mathbf{X}}{\partial \lambda} = - \begin{bmatrix} \boldsymbol{\phi} \\ \mathbf{0}^{\mathsf{T}} \end{bmatrix} \frac{\partial \tilde{\mathbf{X}}_i}{\partial \lambda}.$$

[2] $\because E1 = \underbrace{\frac{\partial \pi_i}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \tilde{\mathbf{X}}_i}}_{\boldsymbol{\phi}} \frac{\partial \tilde{\mathbf{X}}_i}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \lambda} = \boldsymbol{\phi}\mathsf{R}_i \frac{\partial \mathbf{X}}{\partial \lambda}.$

This equation has an exact solution for $\frac{\partial \mathbf{X}}{\partial \lambda}$, so

$$\begin{aligned}
\therefore \frac{\partial \mathbf{X}}{\partial \lambda} &= -\mathsf{R}_i^{-1} \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{N}_i^{\mathsf{T}} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\phi} \\ \mathbf{0}^T \end{bmatrix} \frac{\partial \tilde{\mathbf{X}}_i}{\partial \lambda} \\
&= -\mathsf{R}_i^{-1} \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{N}_i^{\mathsf{T}} \end{bmatrix}^{-1} \left(\begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{N}_i^{\mathsf{T}} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \boldsymbol{N}_i^{\mathsf{T}} \end{bmatrix} \right) \frac{\partial \tilde{\mathbf{X}}_i}{\partial \lambda} \\
&= -\mathsf{R}_i^{-1} \left(\mathbf{I} - \frac{(\vec{\phi_1} \times \vec{\phi_2}) \bigotimes \boldsymbol{N}_i}{\langle (\vec{\phi_1} \times \vec{\phi_2}), \boldsymbol{N}_i \rangle} \right) \frac{\partial \tilde{\mathbf{X}}_i}{\partial \lambda} \\
&= -\mathsf{R}_i^{-1} \left(\mathbf{I} - \frac{\tilde{\mathbf{X}}_i \bigotimes \boldsymbol{N}_i}{\langle \tilde{\mathbf{X}}_i, \boldsymbol{N}_i \rangle} \right) \frac{\partial \tilde{\mathbf{X}}_i}{\partial \lambda} \\
&= -\mathsf{R}_i^{-1} \left(\mathbf{I} - \frac{\tilde{\mathbf{X}}_i \bigotimes \boldsymbol{N}_i}{\langle \tilde{\mathbf{X}}_i, \boldsymbol{N}_i \rangle} \right) \quad (\text{if } \lambda = \mathbf{t}_i).
\end{aligned}$$

where the operator $\bigotimes$ is the outer product operator for vectors. Likewise,

$$\frac{\partial \mathbf{X}}{\partial \lambda} = \mathsf{R}_i^{-1} \left(\mathbf{I} - \frac{\tilde{\mathbf{X}}_i \bigotimes \boldsymbol{N}_i}{\langle \tilde{\mathbf{X}}_i, \boldsymbol{N}_i \rangle} \right) \mathsf{R}_i [\mathbf{X}]_{\mathbf{x}} \frac{\left(\vec{\omega}_i \vec{\omega}_i^{\mathsf{T}} + (\mathsf{R}_i^{\mathsf{T}} - \mathbf{I}) [\vec{\omega}_i]_{\mathbf{x}} \right)}{\|\vec{\omega}_i\|^2} \quad (\text{if } \lambda = \vec{\omega}_i).$$

If $\lambda = \big(L_x, L_y\big)^{\mathsf{T}}$ or $\lambda = \big(\zeta_0, \eta_0\big)^{\mathsf{T}}$, then

$$E2 = \frac{\partial \big(\pi_i(\mathbf{X})\big)}{\partial \lambda} = \frac{\partial}{\partial \lambda} \begin{bmatrix} L_x \frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i} + a \frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \zeta_0 \\ L_y \frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \eta_0 \end{bmatrix},$$

and

$$\boldsymbol{\phi} \mathsf{R}_i \frac{\partial \mathbf{X}}{\partial \lambda} = -\frac{\partial}{\partial \lambda} \begin{bmatrix} L_x \frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i} + a \frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \zeta_0 \\ L_y \frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \eta_0 \end{bmatrix}. \tag{73}$$

Again, adding the condition in equation (72) into equation (73), we get

$$\frac{\partial \mathbf{X}}{\partial \lambda} = -\mathsf{R}_i^{-1} \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{N}_i^{\mathsf{T}} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial \left(\begin{bmatrix} L_x \frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i} + a \frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \zeta_0 \\ L_y \frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i} + \eta_0 \end{bmatrix} \right)}{\partial \lambda} \\ 0 \end{bmatrix}. \tag{74}$$

Based on equation (74), we obtained

$$\frac{\partial \mathbf{X}}{\partial \lambda} = \begin{cases} \frac{1}{\tilde{\mathsf{Z}}_i |\Phi|} \mathsf{R}_i^{-1} \begin{bmatrix} \tilde{\mathsf{X}}_i (\boldsymbol{N}_i \times \vec{\phi_2}) & \tilde{\mathsf{Y}}_i (\vec{\phi_1} \times \boldsymbol{N}_i) \end{bmatrix} & \text{if } \lambda = \big(L_x, L_y\big)^{\mathsf{T}} \\ \frac{1}{|\Phi|} \mathsf{R}_i^{-1} \begin{bmatrix} (\boldsymbol{N}_i \times \vec{\phi_2}) & (\vec{\phi_1} \times \boldsymbol{N}_i) \end{bmatrix} & \text{if } \lambda = \big(\zeta_0, \eta_0\big)^{\mathsf{T}} \end{cases}. \tag{75}$$

Table 8: $\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda}$ and $\frac{\partial(\mathbf{X})}{\partial\lambda}$ with respect to different types of camera parameters

	$\frac{\partial\left(\pi_i(\mathbf{X})\right)}{\partial\lambda}$	$\frac{\partial(\mathbf{X})}{\partial\lambda}$
$\lambda = \left(L_x, L_y\right)^{\mathsf{T}}$	$\frac{1}{\tilde{\mathsf{Z}}_i}\begin{bmatrix}\tilde{\mathsf{X}}_i & 0\\ 0 & \tilde{\mathsf{Y}}_i\end{bmatrix}$	$\frac{1}{\tilde{\mathsf{Z}}_i\lvert\Phi\rvert}\mathsf{R}_i^{-1}\begin{bmatrix}\tilde{\mathsf{X}}_i(\boldsymbol{N}_i\times\vec{\phi_2}) & \tilde{\mathsf{Y}}_i(\vec{\phi_1}\times\boldsymbol{N}_i)\end{bmatrix}$
$\lambda = \left(\zeta_0, \eta_0\right)^{\mathsf{T}}$	$\mathbf{I}$	$\frac{1}{\lvert\Phi\rvert}\mathsf{R}_i^{-1}\begin{bmatrix}(\boldsymbol{N}_i\times\vec{\phi_2}) & (\vec{\phi_1}\times\boldsymbol{N}_i)\end{bmatrix}$
$\lambda = \mathbf{t}_i$	$\boldsymbol{\phi}$	$-\mathsf{R}_i^{-1}\left(\mathbf{I}-\frac{\tilde{\mathbf{X}}_i\otimes\boldsymbol{N}_i}{\langle\tilde{\mathbf{X}}_i,\boldsymbol{N}_i\rangle}\right)$
$\lambda = \vec{\omega}_i$	$-\boldsymbol{\phi}\mathsf{R}_i[\mathbf{X}]_{\mathtt{x}}\Psi$	$\mathsf{R}_i^{-1}\left(\mathbf{I}-\frac{\tilde{\mathbf{X}}_i\otimes\boldsymbol{N}_i}{\langle\tilde{\mathbf{X}}_i,\boldsymbol{N}_i\rangle}\right)\mathsf{R}_i[\mathbf{X}]_{\mathtt{x}}\Psi$

A.3 Summary

The long and complicated derivations above indicate that to refine a specific type of camera parameter, we should update the type of camera parameter using the vector as follows:

$$\begin{aligned}\Delta\lambda \;=\; -\Delta t\;\Bigg\{ & \frac{2}{A_{R_i}}\int_U w_i\Big[w_i f\big(\mathbf{X}\big)+\gamma_i-I_i\big(\pi_i(\mathbf{X})\big)\Big]\left\langle\nabla_{\mathtt{s}} f\big(\mathbf{X}\big),\frac{\partial(\mathbf{X})}{\partial\lambda}\right\rangle J_i d\boldsymbol{u} \\ - \quad & A_{R_i}^{-2}\int_U\Big[I_i\big(\pi_i(\mathbf{X})\big)-\big(w_i f\big(\mathbf{X}\big)+\gamma_i\big)\Big]^2 J_i d\boldsymbol{u}\int_{\partial U}\left\langle Q\frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial s},\frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial\lambda}\right\rangle ds \quad (76)\\ + \quad & \frac{1}{A_{R_i}}\int_{\partial U}\Big[I_i\big(\pi_i(\mathbf{X})\big)-\big(w_i f\big(\mathbf{X}\big)+\gamma_i\big)\Big]^2\left\langle Q\frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial s},\frac{\partial\big(\pi_i(\mathbf{X})\big)}{\partial\lambda}\right\rangle ds\Bigg\}^{\mathsf{T}}, \quad (77)\end{aligned}$$

in which Δt (a scalar) is a positive time step size. Depending on the type of parameter tended to be refined, two corresponding elements are selected from Table 8 for their roles in the equations above.

The symbols used in the table are summarized as follows:

Table 9: Symbols used in Table 8

$\boldsymbol{N}$:	an unit normal observed in the world coordinate frame
$\boldsymbol{N}_i$:	the observation of unit normal $\boldsymbol{N}$ in the i^{th} camera coordinate frame
$\mathbf{I}$:	an identity matrix. $\mathbf{I}$ can be a three-by-three or two-by-two matrix
$\boldsymbol{\phi}$:	$\begin{pmatrix} L_x & a \\ 0 & L_y \end{pmatrix} \begin{pmatrix} \frac{1}{\tilde{\mathsf{Z}}_i} & 0 & -\frac{\tilde{\mathsf{X}}_i}{\tilde{\mathsf{Z}}_i^2} \\ 0 & \frac{1}{\tilde{\mathsf{Z}}_i} & -\frac{\tilde{\mathsf{Y}}_i}{\tilde{\mathsf{Z}}_i^2} \end{pmatrix}$
$\vec{\phi}_1$:	the first row of $\boldsymbol{\phi}$
$\vec{\phi}_2$:	the second row of $\boldsymbol{\phi}$
Φ:	$\begin{bmatrix} \boldsymbol{\phi}^{\mathsf{T}} & \mathbf{N}_i \end{bmatrix}^{\mathsf{T}}$
$[\mathsf{a}]_{\mathbf{x}}$:	$\begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$ given $\mathsf{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^{\mathsf{T}}$
$\vec{\omega}_i$:	the rotation axis of R_i
$\bigotimes$:	the outer product operator of vectors
Ψ:	$\frac{\left(\vec{\omega}_i\vec{\omega}_i^{\mathsf{T}} + (\mathsf{R}_i^{\mathsf{T}} - \mathbf{I})[\vec{\omega}_i]_{\mathbf{x}}\right)}{\|\vec{\omega}_i\|^2}$

APPENDIX B

INEXACT LINE SEARCH

Selecting an appropriate numerical method to locate a local minimum around the initial estimate is an important issue after the gradient descent directions are analytically determined in appendix A. During the development of this research project, many numerical methods, basic and advanced, were adopted to produce a suitable step size along a given descent direction to locate a local minimum. We usually need to provide an upper bound to confine a range in which a local minimum may be found before using these methods. This upper bound is heuristically-determined by our previous experiments and is presented in equation (25). Considering the robustness and convergence rate, we eventually selected the strong Wolfe conditions and the corresponding implementation since they do not demand high computation costs and can produce a moderate estimation of a local minimum. In this section, we particularly emphasize the details of the strong Wolfe conditions. However, we first introduce another method that is simpler than the strong Wolfe conditions—the Armijo rule.

B.1 The Armijo rule

In the following context, the objective function to be minimized with respect to λ is denoted by $E(\lambda)$. Given a descent direction $\vec{p}$ obtained at the initial estimate λ_0, the Armijo rule, an inexact line search method, produces a scalar α so that $E(\lambda_0 + \alpha\vec{p})$ has sufficient decrease in E [29, 30]. Note that $\vec{p}$ does not need to be the gradient descent direction. In practice, $\vec{p}$ can be represented as $\vec{p} = -M^{-1}\nabla E$, in which M is a nonsingular and symmetric matrix. If the gradient descent method is used to decrease E, then M is selected as an identity matrix; by contrast, if Newton's method

is adopted to decrease E, then M is chosen to be the Hessian of E.

Denote

$$F(\alpha) = E(\lambda_0 + \alpha\vec{p})$$

and

$$\hat{F}(\alpha) = F(0) + \epsilon\alpha\frac{\partial E}{\partial \vec{p}} = F(0) + \epsilon\alpha\nabla E^{\mathsf{T}}\vec{p} = F(0) + \epsilon\alpha F'(0), \tag{78}$$

the Armijo rule states that $\alpha_i\vec{p}$ is treated as an acceptable vector to approach a local minimum from the initial estimate, λ_0, if

$$F(\alpha_i) \leq \hat{F}(\alpha_i). \tag{79}$$

The implementation of the Armijo rule is based on the concept of the following backtracking search strategy:

$$\begin{cases} F(\alpha_i) \geq \hat{F}(\alpha_i) \\ F(\alpha_{i+1}) \leq \hat{F}(\alpha_{i+1}) \end{cases}, \tag{80}$$

which is illustrated in Figure 35. Initially, α_0 is set to a large positive scalar, preferably a heuristically-determined upper bound, and then iteratively scaled by $\frac{1}{\varsigma}$ ($\varsigma > 1$) until both inequalities in (80) are satisfied. With an appropriately-chosen ϵ and a large initial estimate of α—α_0, the first inequality of (80) holds, but the second inequality does not; they will both hold after α is gradually reduced to a certain value (denoted by α_{i+1}). At this point, the Armijo rule determines α_{i+1} as an appropriate step size along the descent direction, $\vec{p}$, to reach a local minimum.

Note that ϵ is a scalar whose value is larger than 0 and smaller than 1; that is, $0 < \epsilon < 1$. The function of ϵ can be understood from the demonstration of Figure 35: ϵ not only controls how close an estimated step size can approach a local minimum but also determines the robustness and efficiency of the Armijo rule. The rule of thumb for selecting a good ϵ is to adjust ϵ so that only one local minimum exists in the range of $\alpha = 0$ and $\alpha = \hat{\alpha}$, where $\hat{F}(\hat{\alpha}) = F(\hat{\alpha})$.

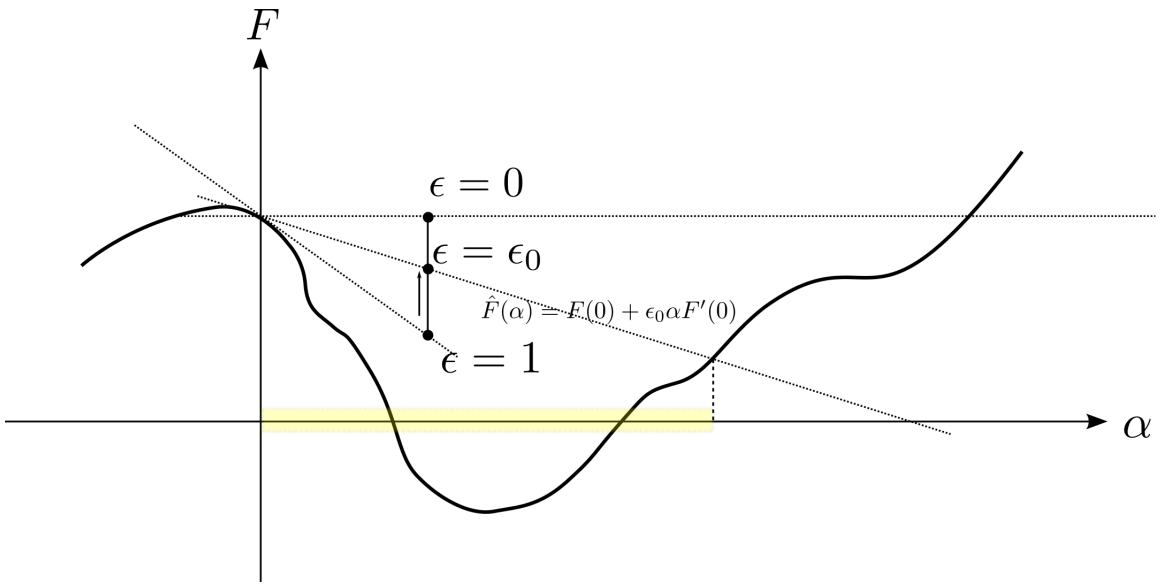


Figure 35: A line search problem for finding a local minimum is effectively an univariate optimization problem, and the Armijo rule provides an inexact but acceptable solution with moderate computation costs spent. Here, $E(\lambda)$ is an objective function, $F(\alpha) = E(\lambda_0 + \alpha\vec{p})$, and $\hat{F}(\alpha) = F(0) + \epsilon\alpha\nabla E^{\mathsf{T}}\vec{p} = F(0) + \epsilon\alpha F'(0)$, where $\vec{p}$ is a descent direction. $\alpha_i\Big|_{i=0}$ is initially set large and iteratively reduced by $\alpha_{i+1} = \frac{\alpha_i}{\varsigma}$ until $\begin{cases} F(\alpha_i) \geq \hat{F}(\alpha_i) \\ F(\frac{\alpha_i}{\varsigma}) \leq \hat{F}(\frac{\alpha_i}{\varsigma}) \end{cases}$ is satisfied. In this figure, any α in the highlighted region can be chosen by the Armijo rule as an appropriate step size for producing sufficient decrease in E.

In a nutshell, the most obvious advantage of the Armijo rule is its simplicity. Other than the sequential evaluations of F, the computation costs required by other parts are low. If ϵ, ς, and α_0 are adequately chosen, the Armijo rule seldom takes too many iterations to stop.

B.2 The strong Wolfe conditions

In many optimization applications, the Armijo rule generates an over-broad range of acceptable step sizes. As seen in Figure 35, points in the highlighted region are treated by the Armijo rule as qualified step sizes for sufficiently reducing E, which is not true given that some points are far from the local minimum in the figure. If we are able to narrow the highlighted part down to a region surrounding the local minimum,

then the iterations of performing the Armijo rule can be reduced. As a result, the strong Wolfe conditions are conceived to overcome the aforementioned drawback of the Armijo rule while preserving the original advantages.

Using the notation introduced in the previous section, the Wolfe conditions are listed as follows:

$$F(\alpha_i) \leq \hat{F}(\alpha_i), \tag{81}$$

$$F'(\alpha_i) \geq \rho F'(0), \tag{82}$$

in which $0 < \epsilon < \rho < 1$. Inequality (81) is exactly the Armijo rule. Since inequality (81) is used to find an α that produces a sufficient decrease of E, it is referred to as the sufficient decrease condition of the Wolfe conditions. By contrast, inequality (82) is referred to as the curvature condition and is designed to guarantee that a chosen α is far away enough from the initial estimate of λ. The roles of both inequalities are illustrated in Figure 36, in which l_2 is the tangent line at $F(0)$, l_3 has the same slope $(F'(0))$ as l_2, and the slope of l_4 is $-F'(0)$. Recall that $\vec{p}$ is a descent direction, so $F'(0)$ must be a negative scalar,[1] as illustrated in Figure 36. When $\rho = 1$, possible αs that satisfy the Wolfe conditions range between $[a, c]$, which is highlighted with light yellow in Figure 36.

If (82) is replaced with the following inequality:

$$\Big|F'(\alpha_i)\Big| \leq \rho\Big|F'(0)\Big|, \tag{83}$$

then (81) and (83) are jointly referred to as the strong Wolfe conditions. Inequality (83) in the strong Wolfe conditions applies a stricter condition to restrict the distribution of α than inequality (82) does. Take Figure 36 as an example again, if $\rho = 1$, then possible αs that satisfy the strong Wolfe conditions range within $[a, b]$ because possible αs causing "over-positive" $F'(\alpha)$ are omitted by the absolute value operator in (83).

[1] Because $F'(0) = \nabla E^{\mathsf{T}} \vec{p} < 0$

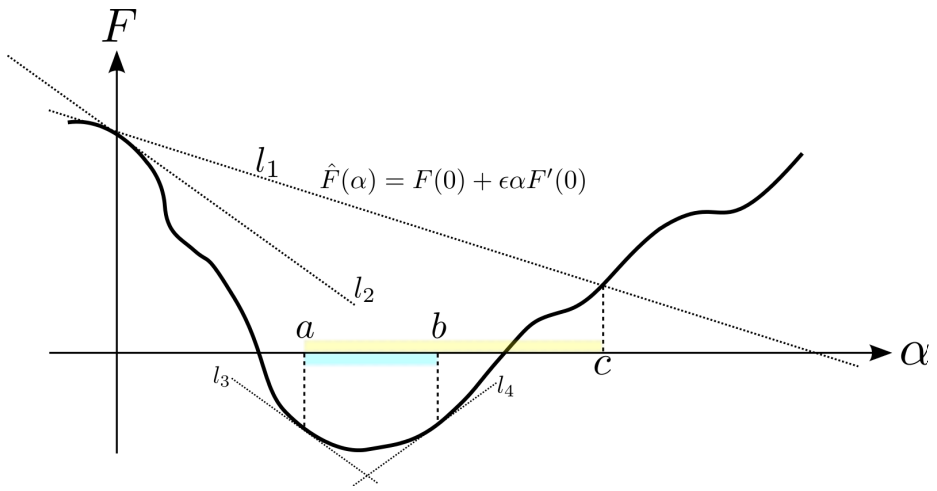


Figure 36: In this figure, the tangent evaluates at $\alpha = 0$ is denoted by l_2. Points within $[a, c]$ are chosen by the Wolfe conditions as appropriate scaling factors of $\vec{p}$ to reach a local minimum, whereas points within $[a, b]$ are selected by the strong Wolfe conditions as appropriate scaling factors.

B.2.1 Implementation of the strong Wolfe conditions

Compared to the implementation of the Armijo rule, the strong Wolfe conditions require more sophisticated implementation. The implementation of the strong Wolfe conditions consists of two phases: bracketing and sectioning, which are thoroughly discussed and rigorously tested in [38, p.60–p.61] and [18, p.34–p.35]. During the bracketing phase, a region in which at least a local minimum exists is identified. Subsequently, the range of the identified region is narrowed down during the sectioning phase. In this section, we depict two phases in Algorithms 1 and 2, respectively, and illustrate them in Figures 37 and 38, respectively. Brief explanations concerning the two phases are given in the following paragraphs.

As described in Algorithm 1, several parameters are set in the beginning of the bracketing phase: $\alpha_{\max}$ can be obtained through equation (25) and α_1 can be obtained through the bisection method, the golden section search, or a polynomial interpolation (discussed in the sectioning phase). Subsequently, a series of tests are conducted on α_1 to estimate its relative position to the confined local minimum, as

Algorithm 1 Bracketing phase of implementing the strong Wolfe conditions

Determine $\alpha_{\max}$, set $\alpha_0 = 0$ and $i = 1$, and estimate $\alpha_1 \in (\alpha_0, \alpha_{\max})$
Loop
    Evaluate $F(\alpha_i)$
    If (81) is not satisfied or if $F(\alpha_i) \geq F(\alpha_{i-1})\cdots$ [1] of Figure 37
        Sectioning (α_{i-1}, α_i)
    Evaluate $F'(\alpha_i)$
    If (83) is satisfied$\cdots$ [2] of Figure 37
        Return the current α_i since it satisfies the strong Wolfe conditions
    If $F'(\alpha_i) \geq 0\cdots$ [3] of Figure 37
        Sectioning (α_i, α_{i-1})
    Estimate $\alpha_{i+1} \in (\alpha_i, \alpha_{\max})\cdots$ [4] of Figure 37
    $i = i + 1$
End loop

elucidated in blocks [1]–[4] in Figure 37. Note that the second-to-last step of the bracketing loop—obtaining the α for the next iteration ($\alpha_{i+1} \in (\alpha_i, \alpha_{\max})$)—can be fulfilled through the same procedure for obtaining α_1.

After the bracketing phase identifies a region containing a local minimum, it passes the lower and upper bounds of the region to the sectioning phase. The two passed arguments are denoted by α_l and α_h, respectively; α_l, the first argument, is a scaling factor determined in the bracketing phase that absolutely satisfies the strong Wolfe conditions. By contrast, α_h, the second argument, is chosen accordingly to be on the opposite side of α_l across the local minimum so that $F'(\alpha_l)(\alpha_h - \alpha_l) < 0$ always holds.[2] With α_l, α_h, and their corresponding information such as evaluations and derivatives available, we can interpolate the information to form a quadratic or a cubic function to approximate F (as shown in [1] of Figure 38). The functions are polynomial, so their critical points can be analytically determined and an adequate one[3] (denoted by α_j) is chosen for further tests. If α_j does not satisfy the sufficient decrease condition, then α_j is assigned to α_h and the algorithm enters the next iteration. On the other hand,

[2]Note that readers should not be confused by the subscripts. α_h does not need to be larger than α_l.

[3]The one that approximates the local minimum

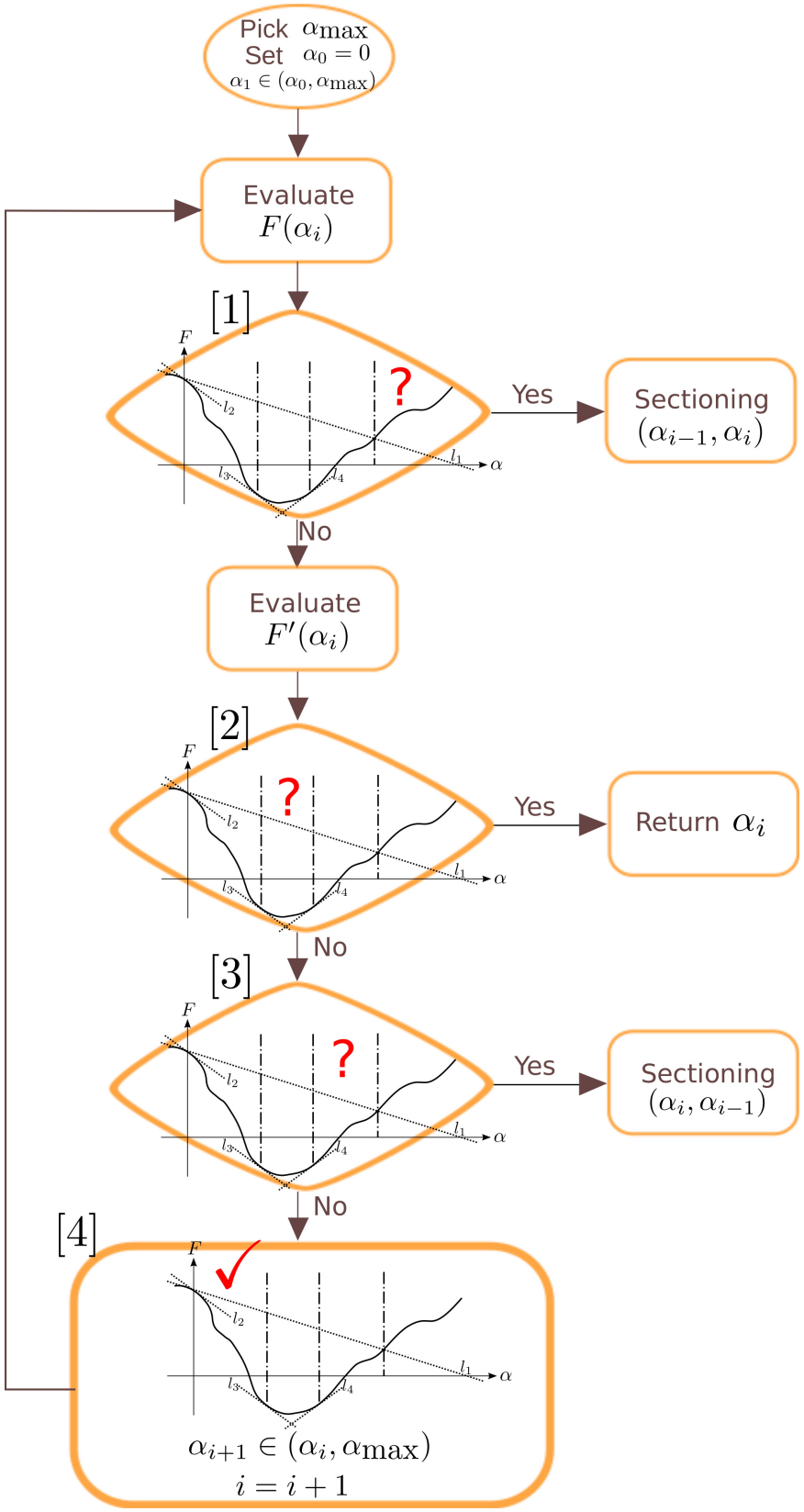


Figure 37: The flow chart and visual illustration of Algorithm1.

Algorithm 2 Sectioning phase of implementing the strong Wolfe conditions

Loop
    Estimate $\alpha_j \in (\alpha_l, \alpha_h)$ through interpolation$\cdots$ [1] of Figure 38
    Evaluate $F(\alpha_j)$
    If (81) is not satisfied or if $F(\alpha_j) \geq F(\alpha_l)\cdots$ [2] of Figure 38
        $\alpha_h = \alpha_j$
    Else
        Evaluate $F'(\alpha_i)$
        If (83) is satisfied$\cdots$ [3] of Figure 38
            Return the current α_j since it satisfies the strong Wolfe conditions
        If $F'(\alpha_j)(\alpha_h - \alpha_l) \geq 0\cdots$ [4] of Figure 38
            $\alpha_h = \alpha_l$
        $\alpha_l = \alpha_j$
    End else
End loop

if α_j does satisfy the sufficient decrease condition, it could either satisfy the curvature condition or not. In the latter case, α_j is assigned to α_l for the next iteration, and α_h is accordingly updated. In all cases, the range of (α_l, α_h) is iteratively reduced through the participation of α_j.

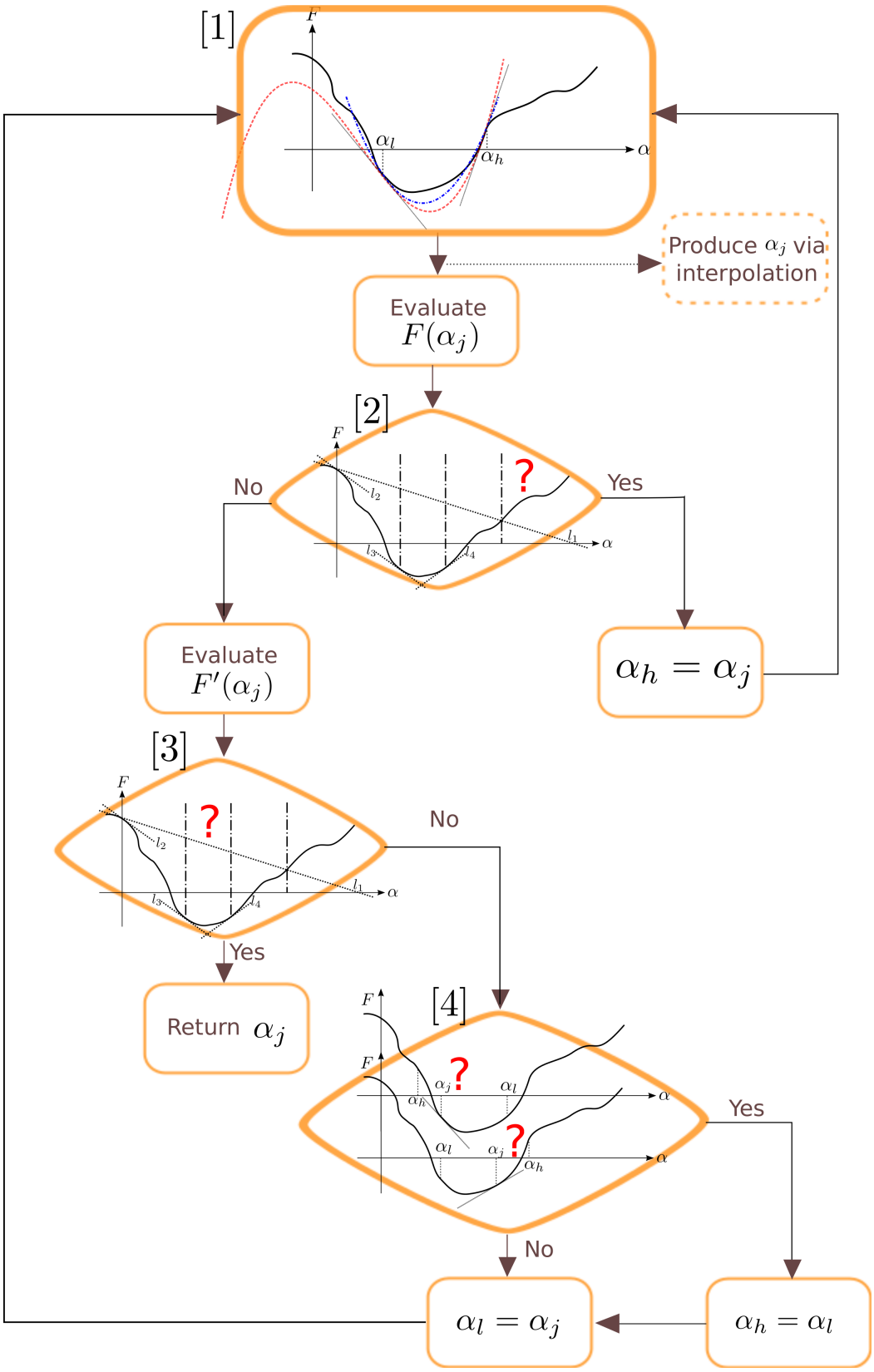


Figure 38: The flow chart and visual illustration of Algorithm2.

APPENDIX C

GRADIENT DESCENT OF THE LOCAL VARIANCE PRIOR

In this section, we derive the gradient descent of E_{cam} with respect to t. E_{cam} is defined as

$$E_{cam} = \frac{\gamma}{2}\int_{\mathbb{T}}\Big\{\int_{\tau-w}^{\tau+w}\frac{\|\lambda-\left({}^{w}\mu^{\tau}\right)\|^2}{2w}dy\Big\}d\tau.$$

In this equation, $\left({}^{w}\mu^{\tau}\right)=\frac{1}{2w}\int_{\tau-w}^{\tau+w}\lambda dy$, so the part inside the curly brackets represents the local variance of λ within the interval of $[\tau-w,\tau+w]$. Taking the derivative of E_{cam} with respect to t, we have

$$\begin{aligned}\frac{\partial E_{cam}}{\partial t} &= \frac{\gamma}{2w}\int_{\mathbb{T}}\int_{\tau-w}^{\tau+w}(\lambda-\left({}^{w}\mu^{\tau}\right))^{\mathsf{T}}(\lambda_t-\left({}^{w}\mu^{\tau}\right)_t)dyd\tau \\ &= \underbrace{\frac{\gamma}{2w}\int_{\mathbb{T}}\int_{\tau-w}^{\tau+w}(\lambda-\left({}^{w}\mu^{\tau}\right))^{\mathsf{T}}\lambda_t dyd\tau}_{(A)}-\underbrace{\frac{\gamma}{2w}\int_{\mathbb{T}}\int_{\tau-w}^{\tau+w}(\lambda-\left({}^{w}\mu^{\tau}\right))^{\mathsf{T}}\left({}^{w}\mu^{\tau}\right)_t dyd\tau}_{(B)}\end{aligned} \quad (84)$$

Term (B) becomes zero based on the following derivation:

$$\begin{aligned}(B) &= \frac{\gamma}{2w}\int_{\mathbb{T}}\int_{\tau-w}^{\tau+w}(\lambda-\left({}^{w}\mu^{\tau}\right))^{\mathsf{T}}\left({}^{w}\mu^{\tau}\right)_t dyd\tau \\ &= \frac{\gamma}{2w}\int_{\mathbb{T}}\int_{\tau-w}^{\tau+w}(\lambda-\left({}^{w}\mu^{\tau}\right))^{\mathsf{T}}\Big[\frac{\partial}{\partial t}\int_{\tau-w}^{\tau+w}\frac{1}{2w}\lambda(x)dx\Big]dyd\tau \\ &= \frac{\gamma}{2w}\int_{\mathbb{T}}\Big[\frac{1}{2w}\int_{\tau-w}^{\tau+w}\lambda_t dx\Big]\cancel{\int_{\tau-w}^{\tau+w}(\lambda-\left({}^{w}\mu^{\tau}\right))^{\mathsf{T}}dyd\tau} \\ &= 0.\end{aligned}$$

Hence,

$$\frac{\partial E_{cam}}{\partial t} = \frac{\gamma}{2w}\int_{\mathbb{T}}\int_{\tau-w}^{\tau+w}(\lambda - ({}^{w}\mu^{\tau}))^{\mathsf{T}}\lambda_t dy d\tau. \quad (85)$$

To further analytically simplify the above representation, we introduce a window function $W^w(\tau, y) = H_{\tau-w}(y) - H_{\tau+w}(y)$, in which H is the Heaviside step function, defined as

$$H_c(y) = \begin{cases} 1 & \text{if } y \geq c \\ 0 & \text{otherwise} \end{cases}.$$

With the window function, equation (85) becomes

$$\begin{aligned} \frac{\partial E_{cam}}{\partial t} &= \frac{\gamma}{2w}\int_{\mathbb{T}}\int_{\tau-w}^{\tau+w}(\lambda - ({}^{w}\mu^{\tau}))^{\mathsf{T}}\lambda_t dy d\tau \\ &= \frac{\gamma}{2w}\Big\{\int_{\mathbb{T}}\int_{\tau-w}^{\tau+w}\lambda^{\mathsf{T}}\lambda_t dy d\tau - \int_{\mathbb{T}}\int_{\tau-w}^{\tau+w}({}^{w}\mu^{\tau})^{\mathsf{T}}\lambda_t dy d\tau\Big\} \\ &= \frac{\gamma}{2w}\Big\{\int_{\mathbb{T}}\int_{\mathbb{T}}\lambda^{\mathsf{T}}\lambda_t W^w(\tau, y) dy d\tau - \int_{\mathbb{T}}\int_{\mathbb{T}}({}^{w}\mu^{\tau})^{\mathsf{T}}\lambda_t W^w(\tau, y) dy d\tau\Big\}. \quad (86) \end{aligned}$$

Since the dual integrals in equation (86) do not depend on each other after the introduction of $W^w(\tau, y)$, we can switch the integrals and rearrange the corresponding terms, yielding

$$
\begin{aligned}
\frac{\partial E_{cam}}{\partial t} &= \frac{\gamma}{2w}\Big\{\int_{\mathbb{T}}\int_{\mathbb{T}}\lambda^{\mathsf{T}}\lambda_t W^w(\tau,y)d\tau dy - \int_{\mathbb{T}}\int_{\mathbb{T}}\big({}^w\mu^\tau\big)^{\mathsf{T}}\lambda_t W^w(\tau,y)d\tau dy\Big\} \\
&= \frac{\gamma}{2w}\Big\{\int_{\mathbb{T}}\lambda_t^{\mathsf{T}}\lambda\int_{\mathbb{T}}W^w(\tau,y)d\tau dy - \int_{\mathbb{T}}\lambda_t^{\mathsf{T}}\int_{\mathbb{T}}\big({}^w\mu^\tau\big)W^w(\tau,y)d\tau dy\Big\} \\
&= \frac{\gamma}{2w}\int_{\mathbb{T}}\lambda_t^{\mathsf{T}}\Big\{\lambda\int_{\mathbb{T}}W^w(\tau,y)d\tau - \int_{\mathbb{T}}\big({}^w\mu^\tau\big)W^w(\tau,y)d\tau\Big\}dy \\
&= \frac{\gamma}{2w}\int_{\mathbb{T}}\lambda_t^{\mathsf{T}}\Big\{\lambda\int_{y-w}^{y+w}d\tau - \int_{y-w}^{y+w}\big({}^w\mu^x\big)dx\Big\}dy \\
&= \frac{\gamma}{2w}\int_{\mathbb{T}}\lambda_t^{\mathsf{T}}\Big\{2w\lambda - \int_{y-w}^{y+w}\underbrace{\big({}^w\mu^x\big)}_{\frac{1}{2w}\int_{x-w}^{x+w}\lambda dz}dx\Big\}dy \\
&= \frac{\gamma}{2w}\int_{\mathbb{T}}\lambda_t^{\mathsf{T}}\Big\{2w\lambda - \frac{1}{2w}\int_{y-w}^{y+w}\left(\int_{x-w}^{x+w}\lambda dz\right)dx\Big\}dy \qquad (87) \\
&= \frac{\gamma}{2w}\int_{\mathbb{T}}\lambda_t^{\mathsf{T}}\Big\{2w\lambda - \frac{1}{2w}\int_{\tau-w}^{\tau+w}\left(\int_{x-w}^{x+w}\lambda dz\right)dx\Big\}d\tau. \qquad (88)
\end{aligned}
$$

Note that ys in equation (87) serve as the temporal variable in the integral, so we can replace ys with τs, the symbol used to denote time variable in the context of this dissertation. After the notation changes, equation (88) can be written as

$$
\frac{\partial E_{cam}}{\partial t} = \left\langle \lambda_t(\tau), \gamma\lambda(\tau) - \tfrac{\gamma}{4w^2}\int_{\tau-w}^{\tau+w}\left(\int_{x-w}^{x+w}\lambda dz\right)dx \right\rangle_{L^2(\mathbb{T})}.
$$

That is , if we use gradient descent to minimize E_{cam} (attaining $\frac{\delta E_{cam}}{\delta\lambda} = 0$), we should set

$$
\lambda_t(\tau) = -\Big\{\gamma\lambda(\tau) - \frac{\gamma}{4w^2}\int_{\tau-w}^{\tau+w}\left(\int_{x-w}^{x+w}\lambda dz\right)dx\Big\}.
$$

REFERENCES

[1] Ahn, S. H., "OpenGL Projection Matrix." Available online at `http://www.songho.ca/opengl/gl\_projectionmatrix\_mathml.html`.

[2] Bechle, A. J. and Wu, C. H., "Virtual wave gauges based upon stereo imaging for measuring surface wave characteristics," *Coastal Engineering*, vol. 58, no. 4, pp. 305–316, 2011.

[3] Benetazzo, A., "Measurements of short water waves using stereo matched image sequences," *Coastal engineering*, vol. 53, no. 12, pp. 1013–1032, 2006.

[4] Benetazzo, A., Fedele, F., Gallego, G., Shih, P., and Yezzi, A., "Offshore stereo measurements of gravity waves," *Coastal engineering*, vol. 64, pp. 127–138, 2012.

[5] Boccotti, P., *Wave mechanics for ocean engineering*, vol. 64. Elsevier oceanography series, 2000.

[6] Bouguet, J., "Camera calibration toolbox for matlab," 2004. Available online at `http://www.vision.caltech.edu/bouguetj/calib\_doc/index.html`.

[7] Bradski, G., "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[8] Briggs, W. L., Henson, V. E., and McCormick, S. F., *A multigrid tutorial*, vol. 72. SIAM, 2000.

[9] Buss, S., *3-D Computer graphics. Mathematical introduction with OpenGL*. Cambridge.

[10] Caselles, V., Kimmel, R., and Sapiro, G., "Geodesic active contours," *International journal of computer vision*, vol. 22, no. 1, pp. 61–79, 1997.

[11] Chan, T. and Vese, L., "Active contours without edges," *Image Processing, IEEE Transactions on*, vol. 10, no. 2, pp. 266–277, 2001.

[12] de Vries, S. and others, "Remote sensing of surf zone waves using stereo imaging," *Coastal Engineering*, vol. 58, no. 3, pp. 239–250, 2011.

[13] Faugeras, O. and Keriven, R., "Variational principles, surface evolution, pde's, level set methods and the stereo problem," *IEEE Trans. Image Processing*, vol. 7, no. 3, pp. 336–344, 1998.

[14] Fedele, F., Gallego, G., Yezzi, A., Benetazzo, A., Cavaleri, L., Sclavo, M., and Bastianini, M., "Euler characteristics of oceanic sea states," *Mathematics and Computers in Simulation*, vol. 82, no. 6, pp. 1102–1111, 2012.

[15] Fedele, F., Benetazzo, A., and Forristall, G. Z., "Space-time waves and spectra in the northern adriatic sea via a wave acquisition stereo system," in *Proc. ASME 30th Int. Conf. on Ocean, Offshore and Arctic Engineering*, 2011.

[16] Fedele, F., Benetazzo, A., Gallego, G., Shih, P.-C., Yezzi, A., Barbariol, F., and Ardhuin, F., "Space-time measurements of oceanic sea states," *Ocean Modelling*, vol. 70, pp. 103–115, 2013.

[17] Fedele, F., Sampath, P., Gallego, G., Yezzi, A., Benetazzo, A., Tayfun, M., Forristall, G. Z., Cavaleri, L., Sclavo, M., and Bastianini, M., "Beyond Waves and Spectra: Euler Characteristics of Oceanic Sea States," in *Proc. ASME 28th Int. Conf. on Ocean, Offshore and Arctic Engineering*, pp. 413–420, American Society of Mechanical Engineers, 2009.

[18] Fletcher, R., *Practical methods of optimization*. John Wiley and Sons, 2000.

[19] Gallego, G., Yezzi, A., Fedele, F., and Benetazzo, A., "A Variational Stereo Method for the Three-Dimensional Reconstruction of Ocean Waves," *Geoscience and Remote Sensing, IEEE Transactions on*, no. 99, pp. 1–13, 2011.

[20] Gallego, G. and Yezzi, A., "A compact formula for the derivative of a 3-D rotation in exponential coordinates," *arXiv preprint arXiv:1312.0788*, 2013.

[21] Gallego Bonet, G., "Variational image processing algorithms for the stereoscopic space-time reconstruction of water waves," 2011.

[22] Gallego Bonet, G., Yezzi, A., Fedele, F., and Benetazzo, A., "Space-time Reconstruction of Oceanic Sea States via Variational Stereo Methods," 2012.

[23] Hartley, R. and Zisserman, A., *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 ed., 2004.

[24] Hartley, R. I., "In defense of the eight-point algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580–593, 1997.

[25] Holthuijsen, L., "Observations of the directional distribution of ocean-wave energy in fetch-limited conditions," *Journal of Physical Oceanography*, vol. 13, no. 2, pp. 191–207, 1983.

[26] Isaacson, E. and Keller, H., *Analysis of numerical methods*. Dover Publications, 1994.

[27] Jin, H., *Variational Methods for Shape Reconstruction in Computer Vision*. PhD thesis, Washington University in St. Louis, Saint Louis, MS, USA, 2003.

[28] Jin, H., Soatto, S., and Yezzi, A., "Multi-view Stereo Beyond Lambert," in *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[29] Kelley, C., *Iterative methods for linear and nonlinear equations*. Society for Industrial and Applied Mathematics, 1995.

[30] Kelley, C., *Iterative methods for optimization*, vol. 18. Society for Industrial and Applied Mathematics, 1999.

[31] Krogstad, H. E. and F Barstow, S., "Satellite wave measurements for coastal engineering applications," *Coastal Engineering*, vol. 37, no. 3, pp. 283–307, 1999.

[32] Liu, P., Schwab, D., Wu, C., and MacHutchon, K., "Wave heights in a 4d ocean wave field," in *Proceedings of the 27th International Conference on Offshore Mechanics and Arctic Engineering*, pp. 15–20, 2008.

[33] MacHutchon, K. and Liu, P., "Measurement and analysis of ocean wave fields in four dimensions," in *OMAE*, vol. 1, pp. 923–927, 2007.

[34] MacHutchon, K. and Liu, P., "Measurement and analysis of ocean wave fields in four dimensions," in *Proceedings of the 26th International Conference on Offshore Mechanics and Arctic Engineering*, 2007.

[35] Makadia, A., Patterson, A., and Daniilidis, K., "Fully automatic registration of 3D point clouds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, pp. 1297–1304, IEEE, 2006.

[36] Morris, D., Kanatani, K., and Kanade, T., "Gauge fixing for accurate 3D estimation," in *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. II–343–II–350, 2001.

[37] Mumford, D. and Shah, J., "Optimal approximations by piecewise smooth functions and associated variational problems," *Communications on pure and applied mathematics*, vol. 42, no. 5, pp. 577–685, 1989.

[38] Nocedal, J. and Wright, S., *Numerical optimization: Springer series in operations research and financial*. Springer, 2 ed., 2006.

[39] NVIDIA Corporation, "NVIDIA CUDA SDK - Physically-Based Simulation." Available online at `http://www.nvidia.com/content/cudazone/cuda\_sdk/Physically-Based\_Simulation.html\#oceanFFT`.

[40] Osher, S. and Paragios, N., *Geometric level set methods in imaging, vision, and graphics*. Springer, 2003.

[41] Santel, F., Heipke, C., Konnecke, S., and Wegmann, H., "Image sequence matching for the determination of three-dimensional wave surfaces," *International archives of photogrammetry remote sensing and spatial information sciences*, vol. 34, no. 5, pp. 596–600, 2002.

[42] Santel, F., Linder, W., and Heipke, C., "Stereoscopic 3D-image sequence analysis of sea surfaces," in *Proceedings of the ISPRS Commission V Symposium*, vol. 35, pp. 708–712, 2004.

[43] Shreiner, D., Sellers, G., Kessenich, J., and Licea-Kane, B., *OpenGL Programming Guide: The Official Guide to Learning OpenGL Version 4.3*.

[44] "The Stanford 3D Scanning Repository." Available online at `http://graphics.stanford.edu/data/3Dscanrep/`.

[45] Sugimori, Y., "A study of the application of the holographic method to the determination of the directional spectrum of ocean waves," vol. 22, no. 5, pp. 339–350, 1975.

[46] Tayfun, M. and Fedele, F., "Wave-height distributions and nonlinear effects," *Ocean engineering*, vol. 34, no. 11, pp. 1631–1649, 2007.

[47] Tessendorf, J., "Simulating Ocean Water," in *SIGGRAPH course notes*, 1999-2004. Available online at `http://jerrytessendorf.blogspot.com.es/`.

[48] Unal, G., Yezzi, A., Soatto, S., and Slabaugh, G., "A variational approach to problems in calibration of multiple cameras," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 8, pp. 1322–1338, 2007.

[49] Wanek, J. and Wu, C., "Automated trinocular stereo imaging system for three-dimensional surface wave measurements," *Ocean engineering*, vol. 33, no. 5, pp. 723–747, 2006.

[50] Worsley, K. J., "The geometry of random images," *Chance*, vol. 9, no. 1, pp. 27–40, 1996.

[51] Yezzi, A. and Soatto, S., "Stereoscopic segmentation," *International Journal of Computer Vision*, vol. 53, no. 1, pp. 31–43, 2003.

[52] Zakharov, V., "Statistical theory of gravity and capillary waves on the surface of a finite-depth fluid," *European journal of mechanics. B, Fluids*, vol. 18, no. 3, pp. 327–344, 1999.

VITA

A native of Changhua, Taiwan, Ping-Chang Shih is a graduate research assistant at the Laboratory of Computational Computer Vision in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. He was awarded "The Excellent Officer" while serving in the Marine Corps, "The Scholarship for Studying Abroad" while pursuing his Ph.D. degree, the GTRIC travel grant after presenting his research poster at the Georgia Tech Research and Innovation Conference, the "NSF Civil, Mechanical and Manufacturing Innovation Conference Graduate Student Fellowship" to attend the conference and present his work, and the "ASME OMAE Outreach Scholarship" to participate in the conference forum. His research focuses on the calibration refinement of a 4-D reconstruction system of ocean surfaces using stereo computer vision principles, variational optimization theory, numerical methods, and high performance computing techniques. In his leisure time, he enjoys science, history, and travel. His wish is to become a novelist.